Methods of Cloud Computing

Beyond Cloud

Complex and Distributed Systems
Faculty IV
Technische Universität Berlin

Operating Systems and Middleware
Hasso-Plattner-Institut
Universität Potsdam

Overview

- Intro
- Federated, Hybrid and Multi-Clouds
  - Concepts
  - Policy enforcement
- Fog, Edge and IoT
  - Concepts
  - Placement in fog architectures
  - Development practices: Testbed for fog computing

Overview

- **Intro**
- Federated, Hybrid and Multi-Clouds
  - Concepts
  - Policy enforcement
- Fog, Edge and IoT
  - Concepts
  - Placement in fog architectures
  - Development practices: Testbed for fog computing

Background

Cluster

- Many computers in one room

Grid

- Loosely coupled computers or clusters all over the world

Cloud

- IT resources as a utility

Edge

- Extending the Cloud to the edge of the network

Cluster

- Mostly homogeneous compute resources and software stacks
- Interconnected by a low-latency and high-bandwidth network
- Goal: improving availability and price/performance

- Examples
  - Analytics cluster at Facebook

Grid

- Heterogeneous compute resources
- Connected via a slow network (i.e. the internet)
- Heterogeneous software stacks unified by a middleware

- Examples
  - PanDA (CERN)
  - BOINC

Background

Cluster

- Many computers in one room

Grid

- Loosely coupled computers or clusters all over the world

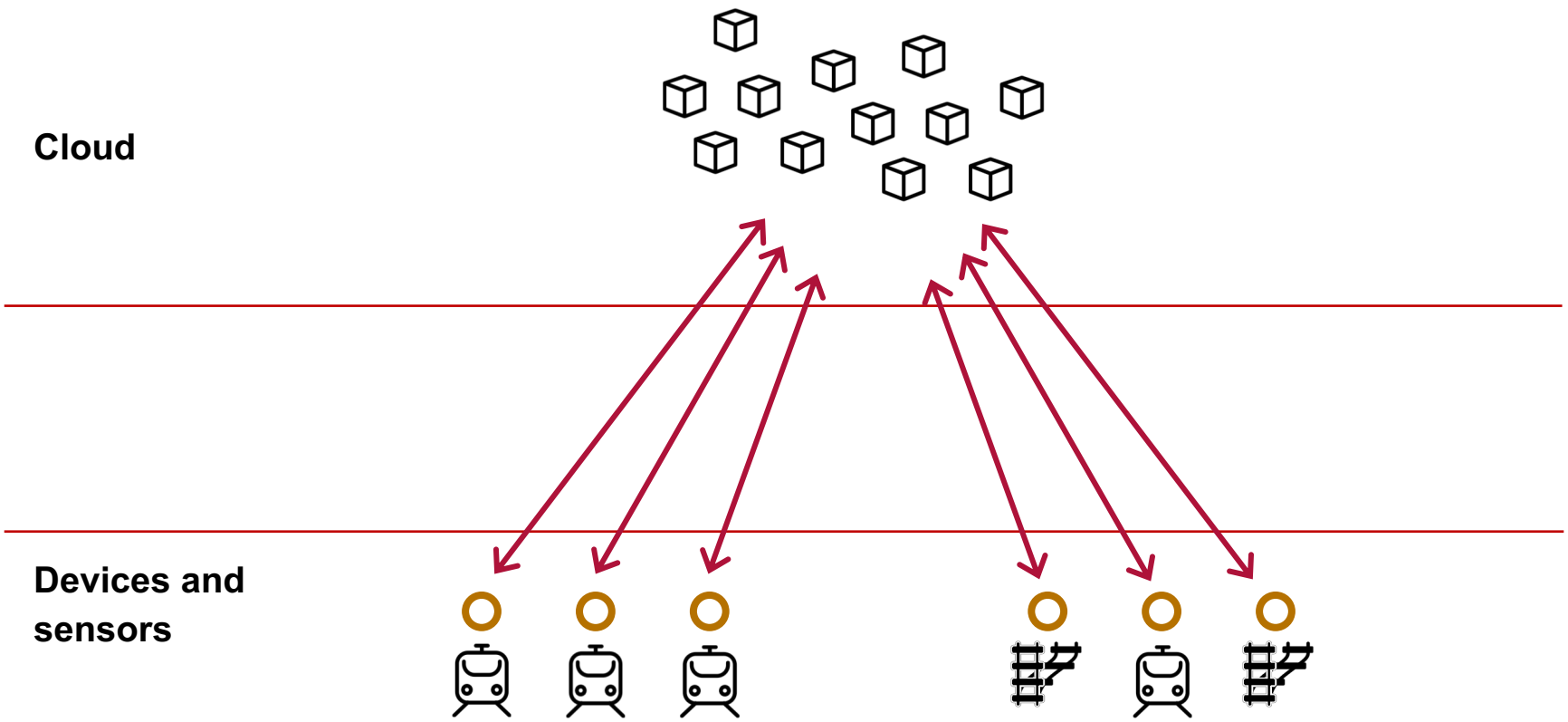
Cloud

- IT resources as a utility

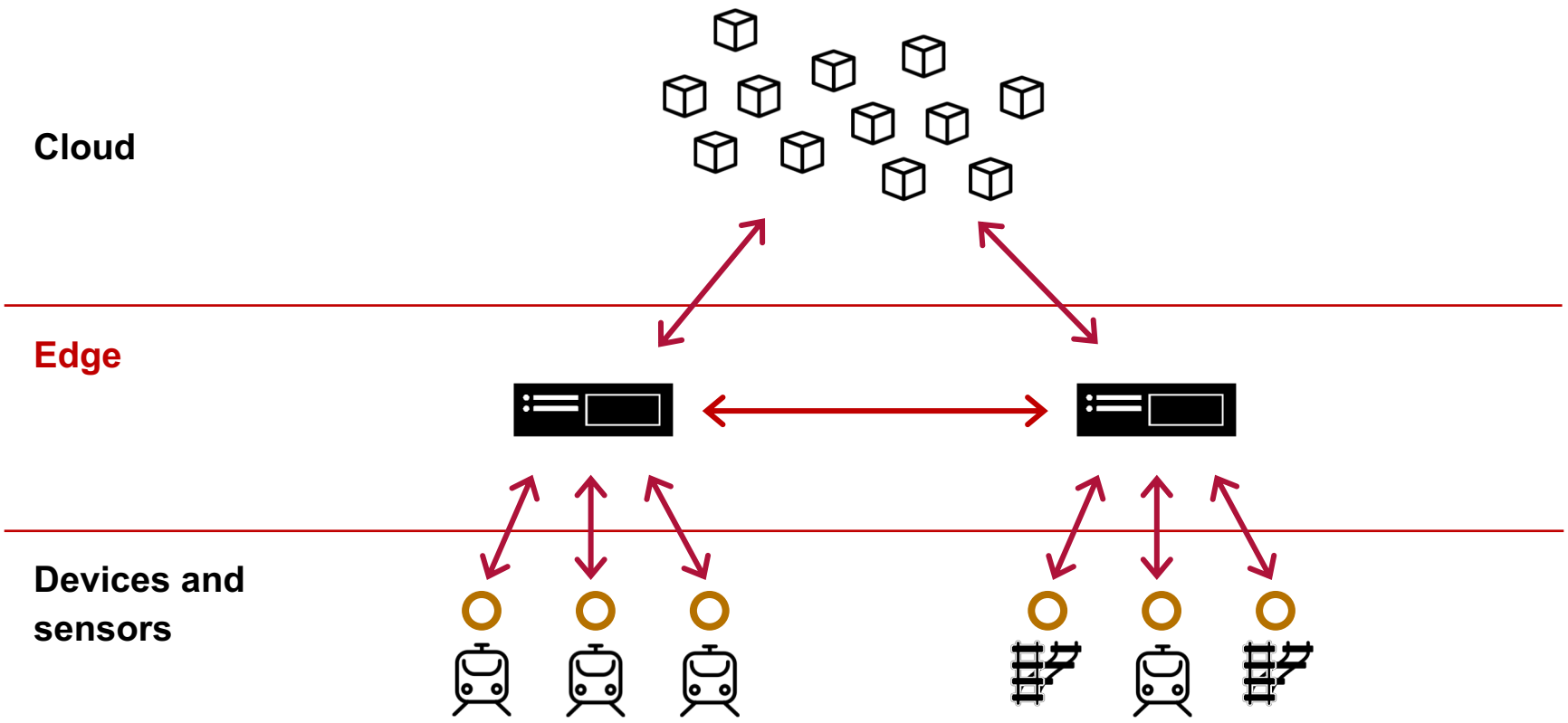
Edge

- Extending the Cloud to the edge of the network

Edge Computing



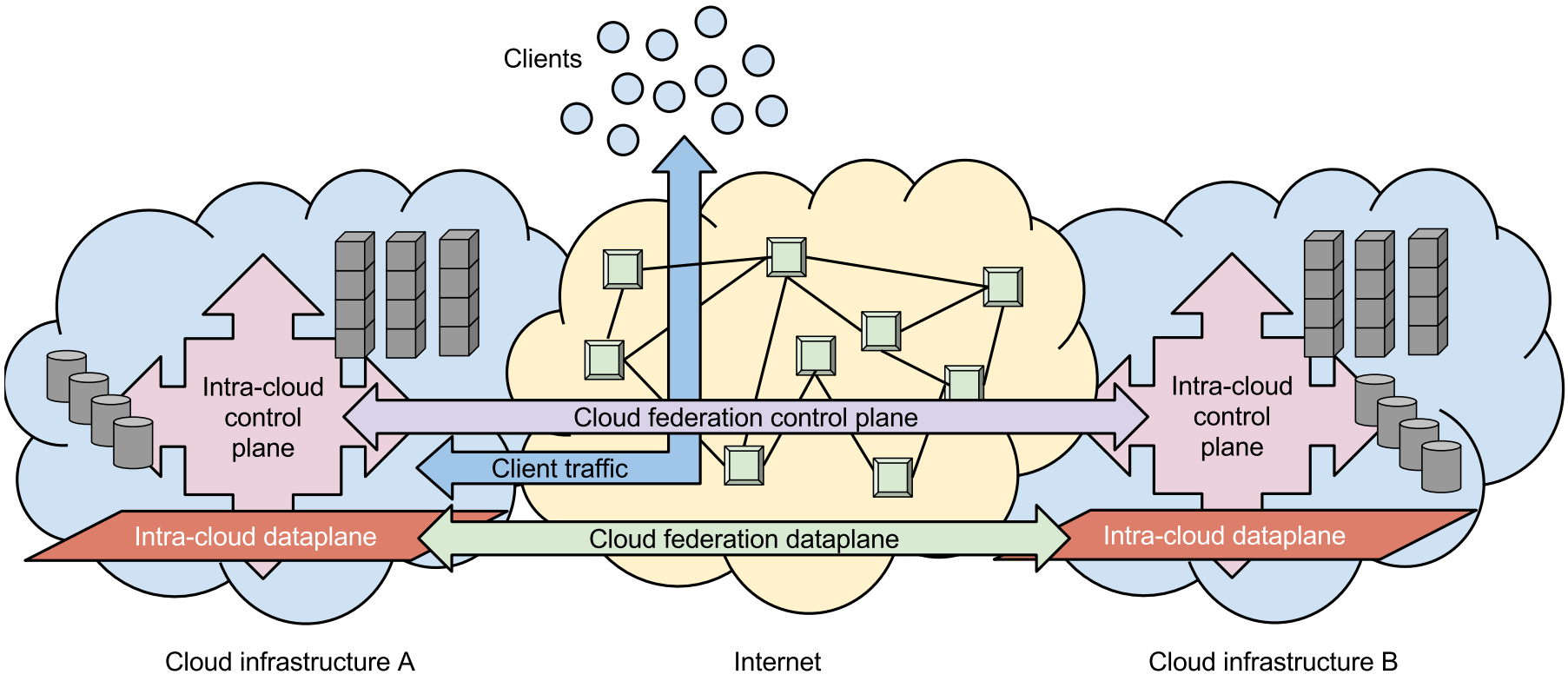
Edge Computing



Overview

- Intro
- **Federated, Hybrid and Multi-Clouds**
  - Concepts
  - Policy enforcement
- Fog, Edge and IoT
  - Concepts
  - Placement in fog architectures
  - Development practices: Testbed for fog computing

Cloud Federation



Overview

- Intro
- Federated, Hybrid and Multi-Clouds
  - **Concepts**
  - Policy enforcement
- Fog, Edge and IoT
  - Concepts
  - Placement in fog architectures
  - Development practices: Testbed for fog computing

Concepts

- Combining providers and models

- Multi-cloud:
  - Using multiple cloud providers in one application
- Federated Cloud:
  - Integrating multiple cloud providers to provide a service
- Hybrid Cloud:
  - Mixing public and private cloud models

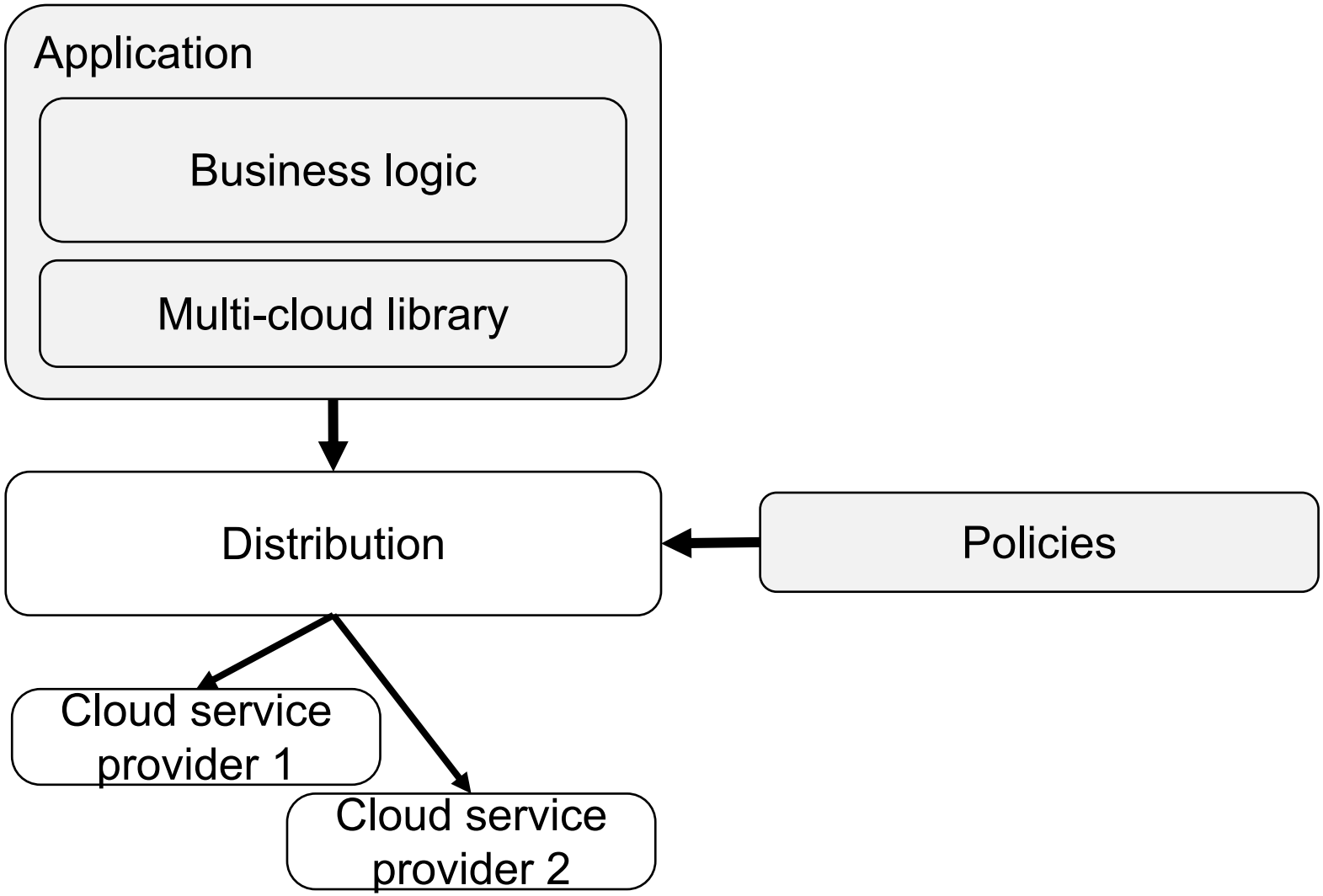
Motivation for Cloud Federation

- Resilience against vendor outages
- Fighting vendor lock-in
- Harness the diversity of cloud services by different providers
- Harness different locations of providers

- Compliance with policies
  - i.e. data needs to be stored in EU

    → Needs a way to express and enforce the policies

Cloud Federation



Overview

- Intro
- Federated, Hybrid and Multi-Clouds
  - Concepts
  - **Policy enforcement**
- Fog, Edge and IoT
  - Concepts
  - Placement in fog architectures
  - Development practices: Testbed for fog computing

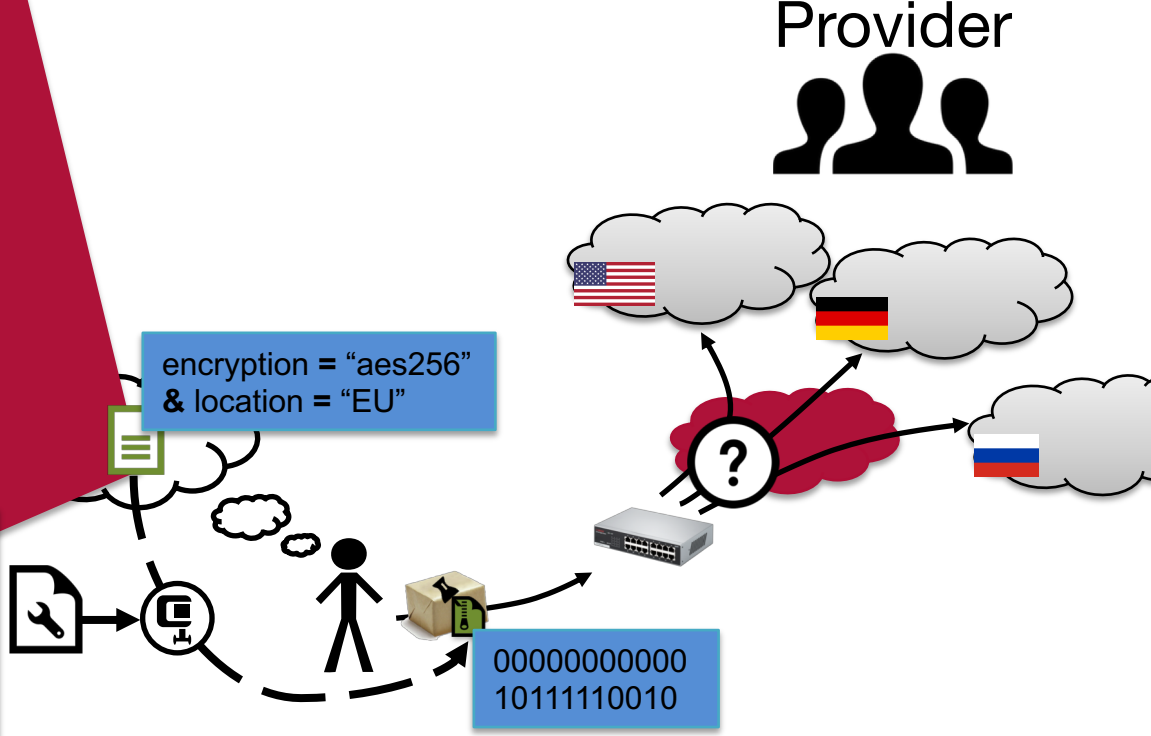
Customer Requirements
CPPL Policy Language

- Location
- Time of deletion
- Notification on access
- Protection
- Replication
- …

- Location
- Type
- …

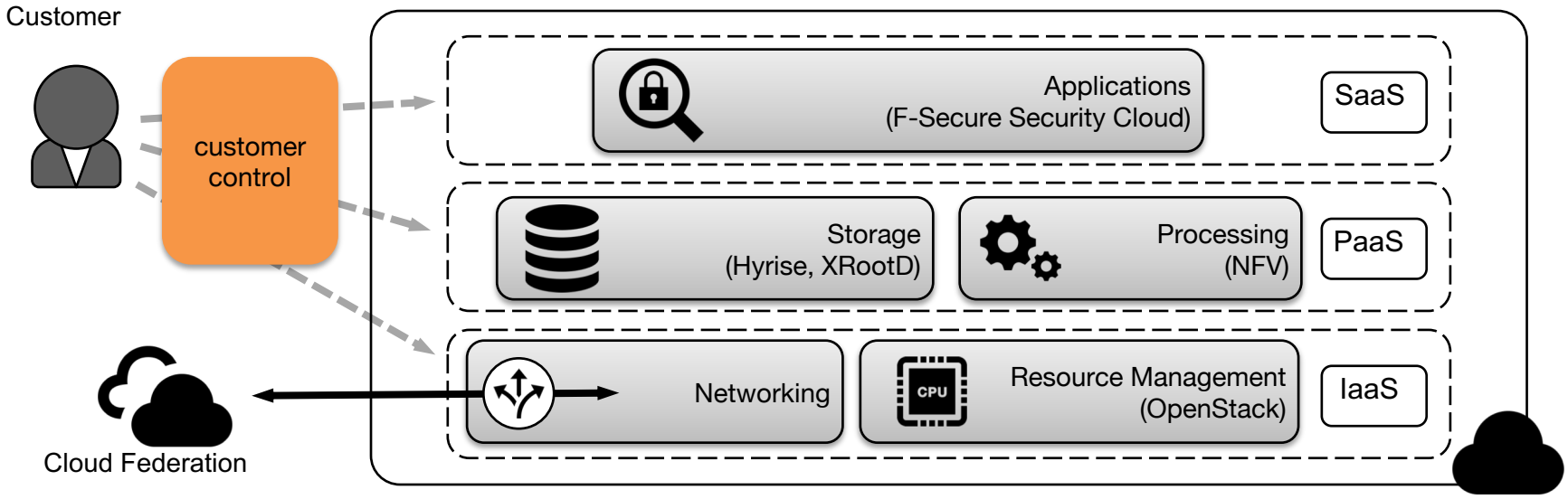
- Restricted paths
- Transport protection
- …

```
{
  “name”: ”encryption”, “type”: “string”,
  “values”: [“aes192”, “aes256”]
},
{
  ”name”: “location”, “type”: “string”,
  “values": [“US”, “DE”, ”RU”, “EU”]
},
```



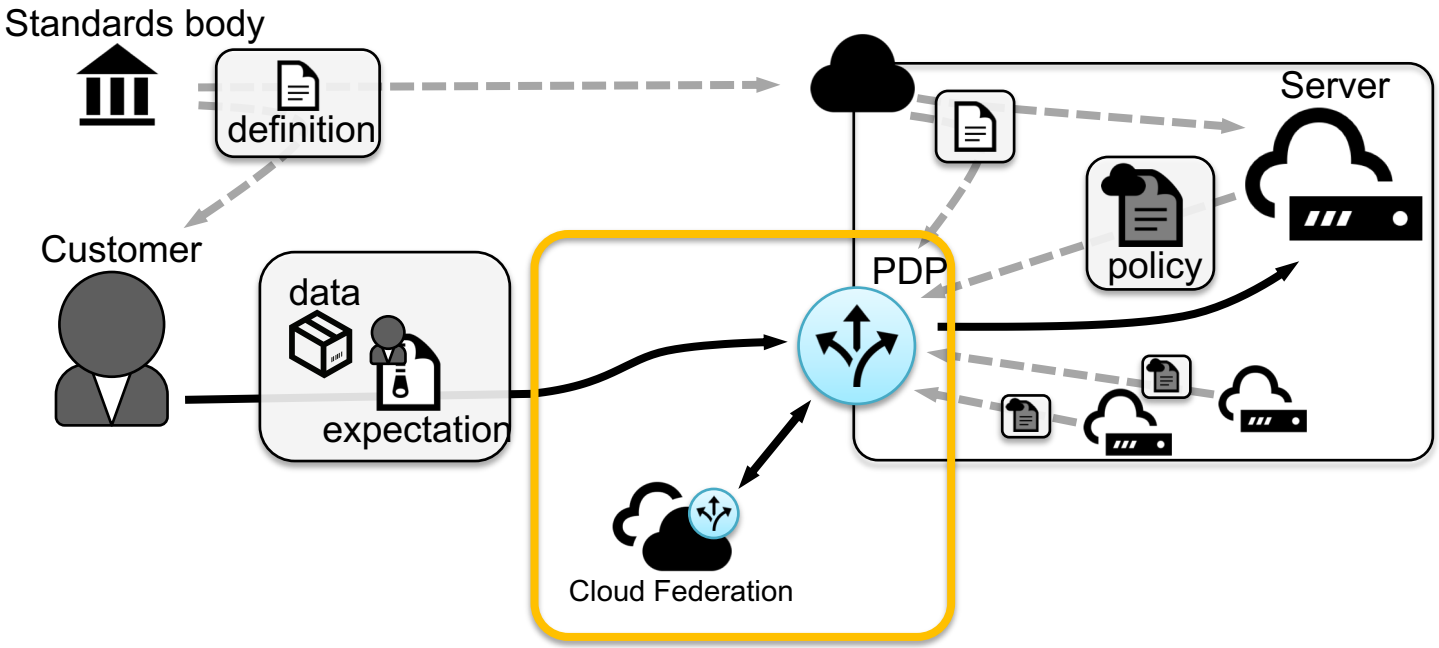
Policy-Aware Cloud Architecture

- Policy-awareness must be addressed at all cloud layers



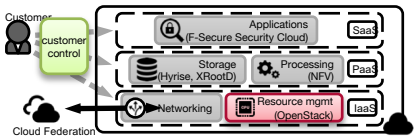
Decisions at the Edge

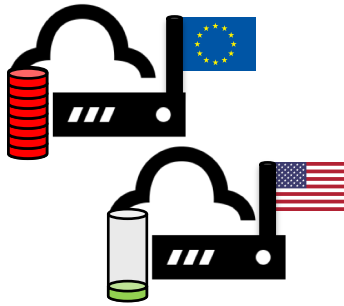
- Policies often restrict set of eligible servers
- Policy decision points:
  - Check policies and assign data to complying systems
  - Realize policy compliance across clouds of a federation

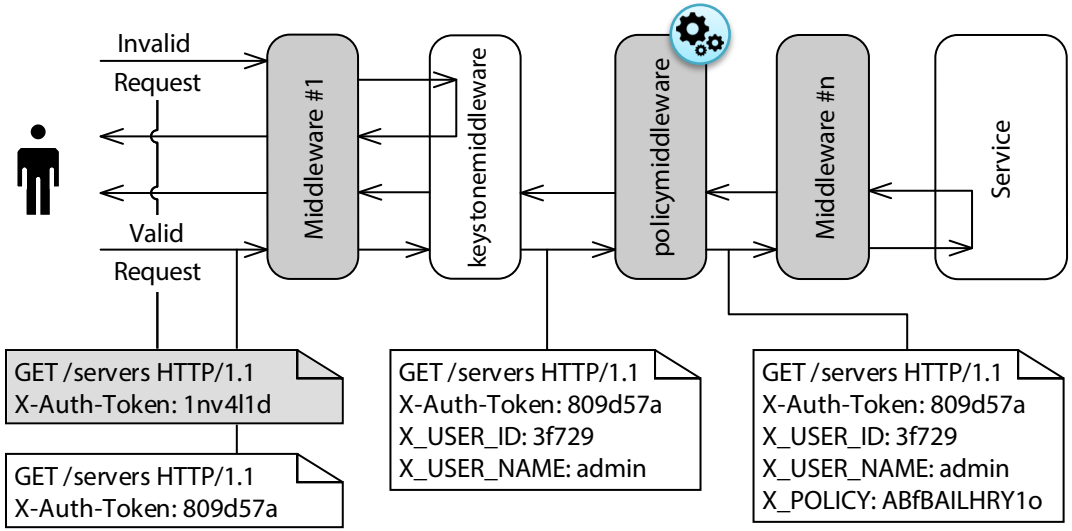


OpenStack – Policy-Aware Resource Management

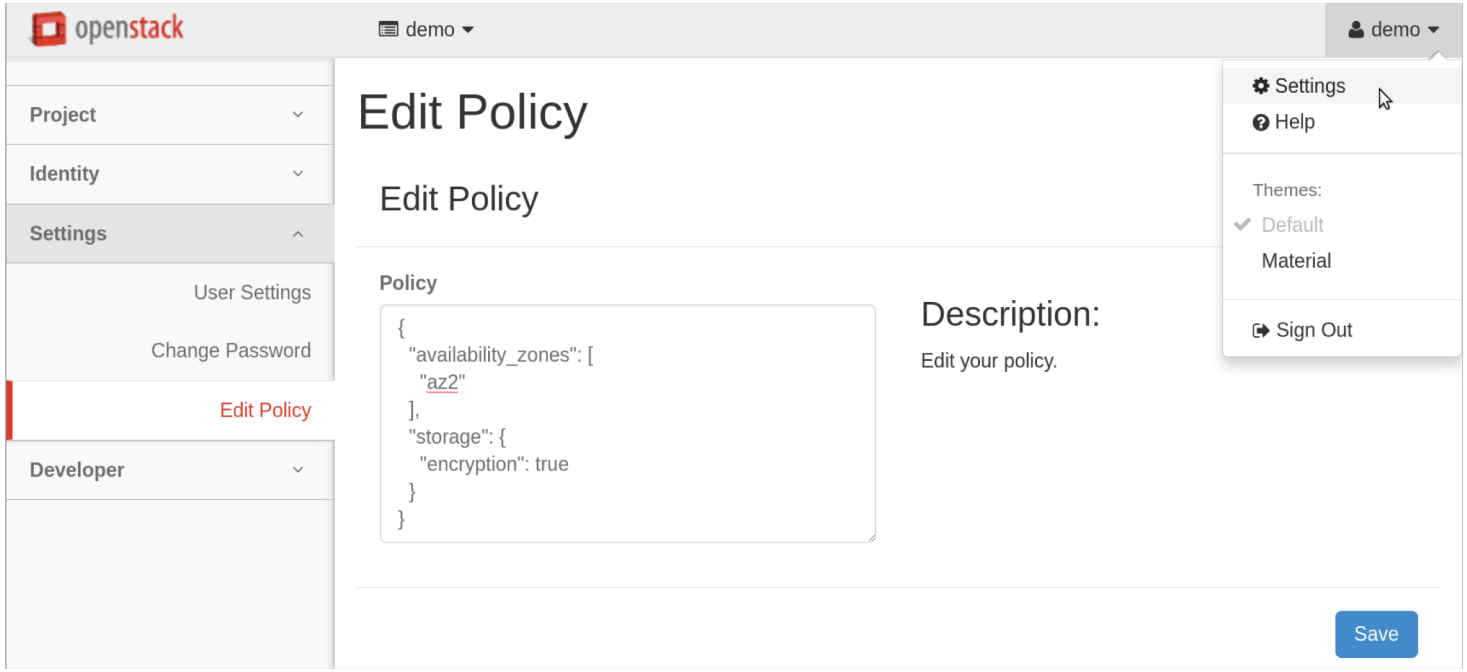
- Elasticity in a policy-aware cloud
  - Often requested but seldom provided attributes → High load
  - Traditional elasticity is **not** policy-aware
- Policy-aware resource management for **OpenStack**



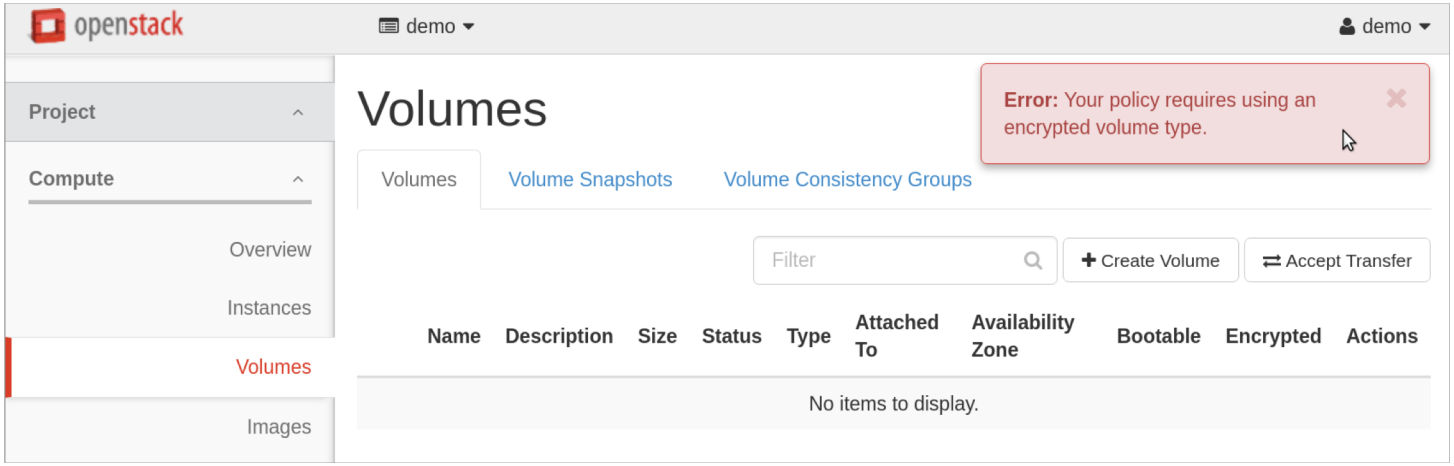


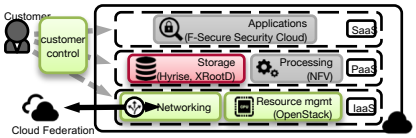


OpenStack – Policy-Aware Resource Management



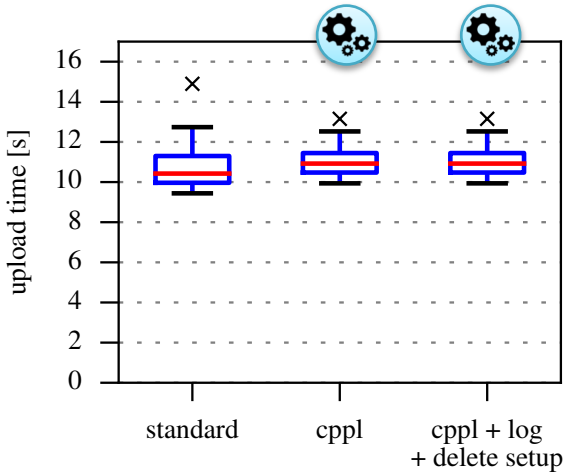
OpenStack – Policy-Aware Resource Management

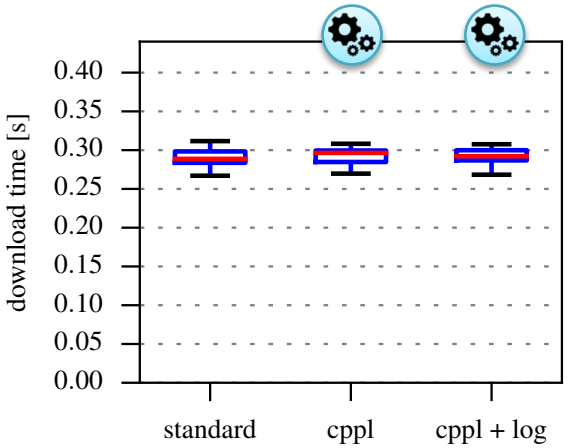


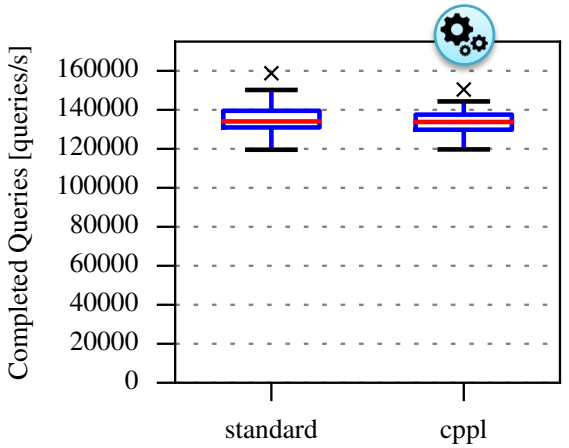


Policy-Aware Storage

- Realized in **XRootD** (CERN) and **Hyrise**
  - Big data
  - High transactional workloads



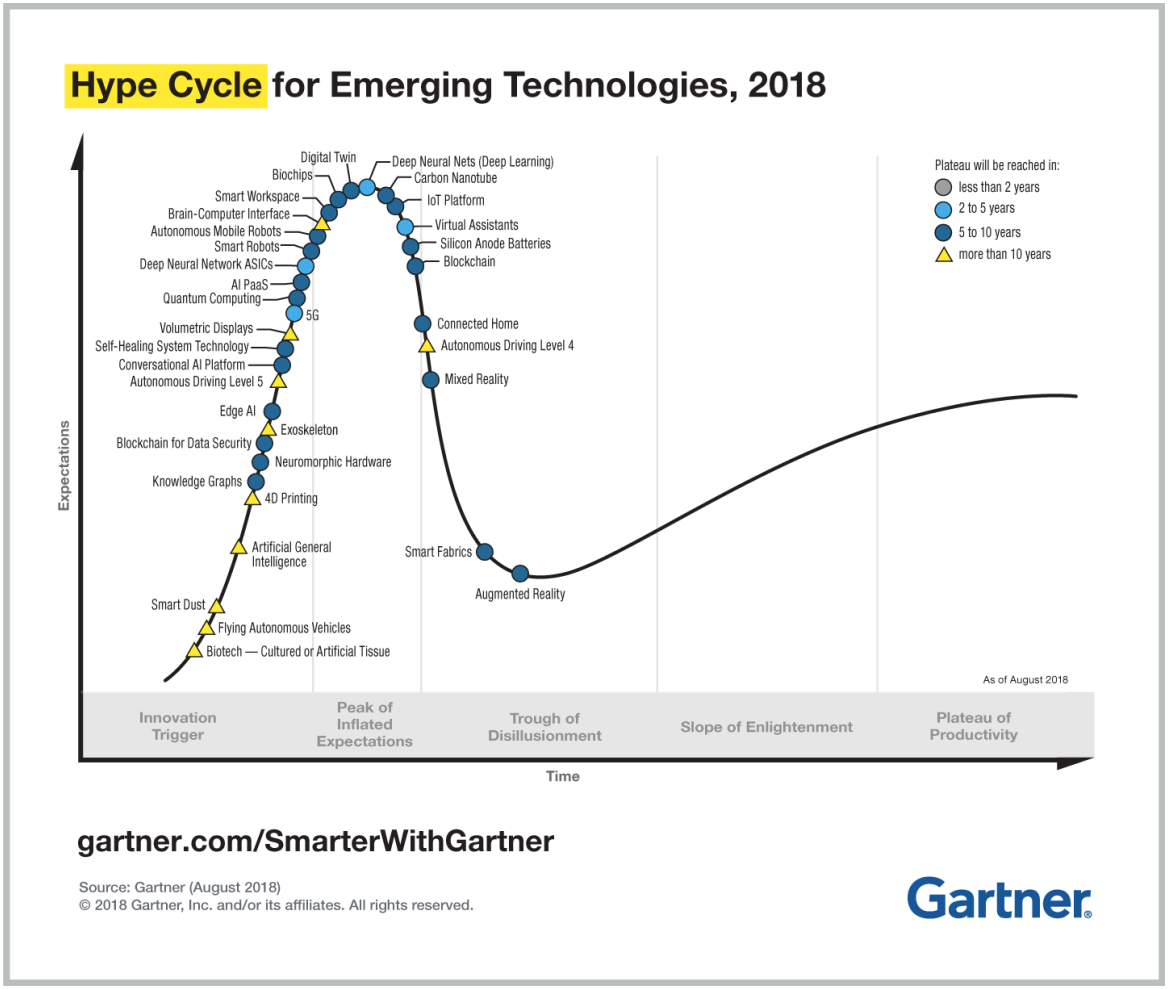




Overview

- Intro
- Federated, Hybrid and Multi-Clouds
  - Concepts
  - Policy enforcement
- **Fog, Edge and IoT**
  - Concepts
  - Placement in fog architectures
  - Development practices: Testbed for fog computing

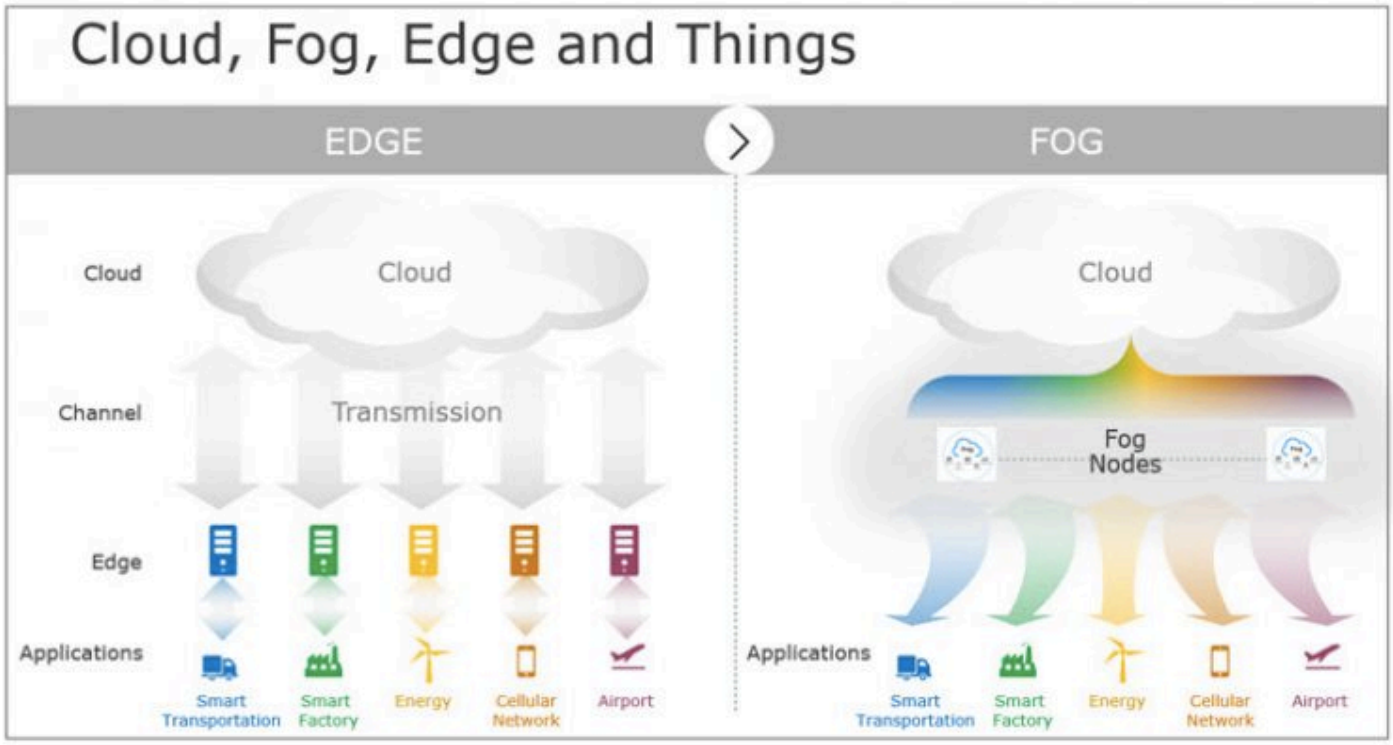
Fog, Edge and IoT



Overview

- Intro
- Federated, Hybrid and Multi-Clouds
  - Concepts
  - Policy enforcement
- Fog, Edge and IoT
  - **Concepts**
  - Placement in fog architectures
  - Development practices: Testbed for fog computing

Edge and Fog



https://www.openfogconsortium.org/

Cloudlets

- Paper from 2009

The Case for VM-based Cloudlets in Mobile Computing

Mahadev Satyanarayanan[†], Paramvir Bahl[‡], Ramon Caceres[•], Nigel Davies[°]
[†]Carnegie Mellon University, [‡]Microsoft Research, [•]AT&T Research, [°]Lancaster University

- Idea: deploy micro data centers (called cloudlets) everywhere for usage in mobile applications
- Geo-distributed like wifi routers
- Term now also describes nodes in a fog architecture (i.e. [1])

Cloudlets

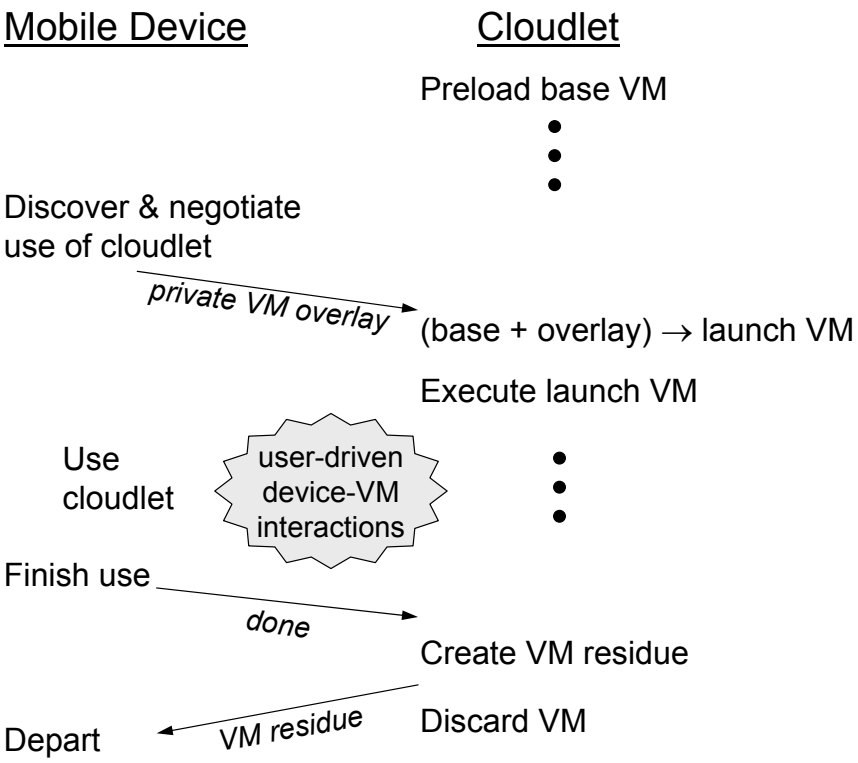
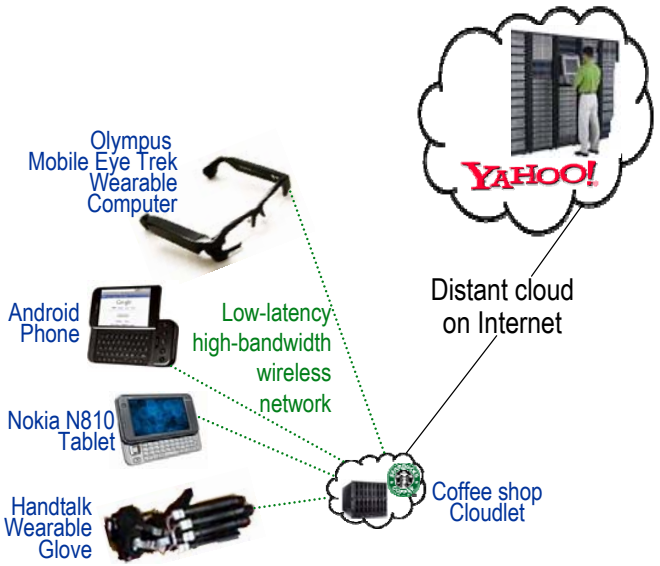


Figure 5: Dynamic VM Synthesis Timeline

Cloudlets and Clouds



(a) Cloudlet Concept

	Cloudlet	**Cloud**
State	Only soft state	Hard and soft state
Management	Self-managed; little to no professional attention	Professionally administered, 24x7 operator
Environment	"Datacenter in a box" at business premises	Machine room with power conditioning and cooling
Ownership	Decentralized ownership by local business	Centralized ownership by Amazon, Yahoo!, etc.
Network	LAN latency/bandwidth	Internet latency/bandwidth
Sharing	Few users at a time	100s-1000s of users at a time

(b) Key Differences: Cloudlet vs. Cloud

Cloudlets: Proof of Concept Architecture

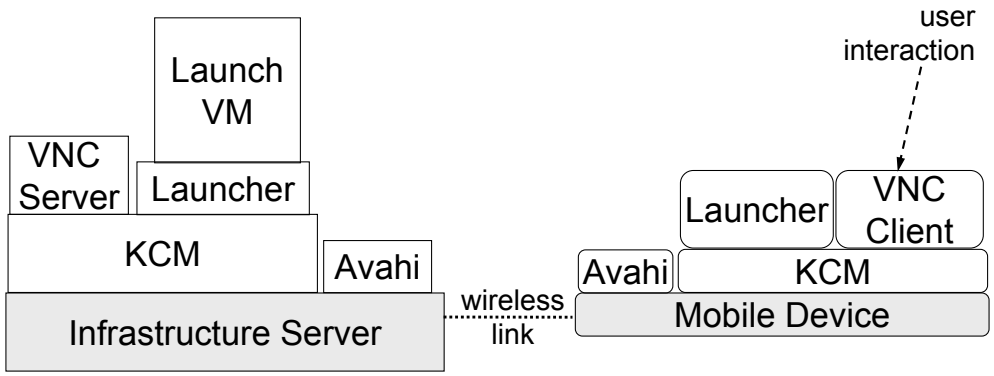


Figure 6: Runtime Binding in Kimberley

Overview

- Intro
- Federated, Hybrid and Multi-Clouds
  - Concepts
  - Policy enforcement
- Fog, Edge and IoT
  - Concepts
  - **Placement in fog architectures**
  - Development practices: Testbed for fog computing

Scheduling in Fog Architectures

- Workshop paper from 2018

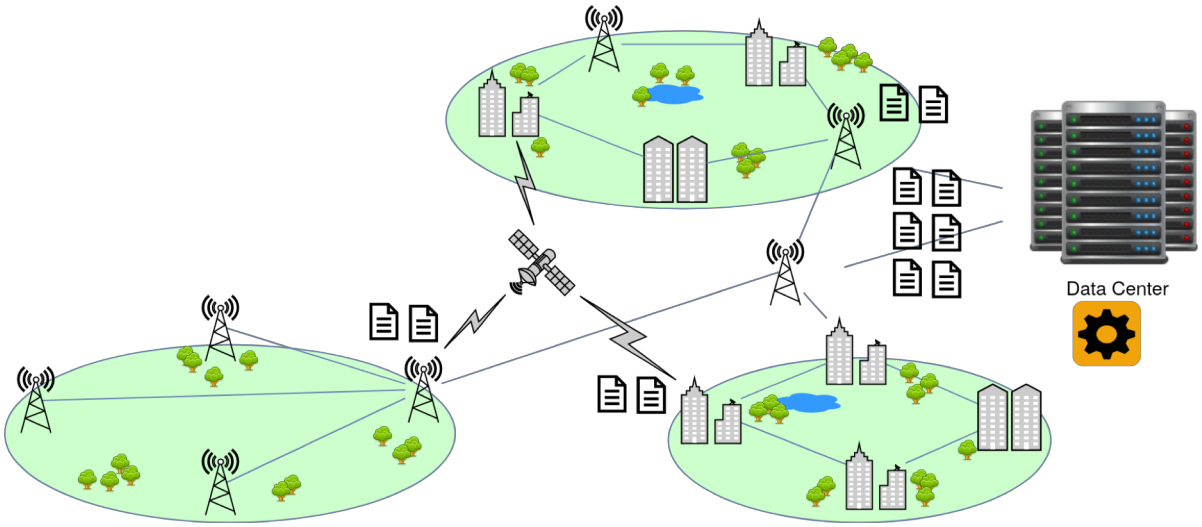
Scheduling Stream Processing Tasks on
Geo-Distributed Heterogeneous Resources

Gerrit Janßen, Ilya Verbitskiy, Thomas Renner, and Lauritz Thamsen
Technische Universität Berlin, Germany
{firstname.lastname}@tu-berlin.de

- Work done within the Berlin Big Data Center (BBDC), in Phase 2 on Distr. Processing of IoT Data Streams
- Empirical evaluation of different methods for scheduling Flink tasks across heterogeneous distributed nodes

Data Streams of the IoT

- Various sensors deployed across factories and urban infrastructures, continuously recording data

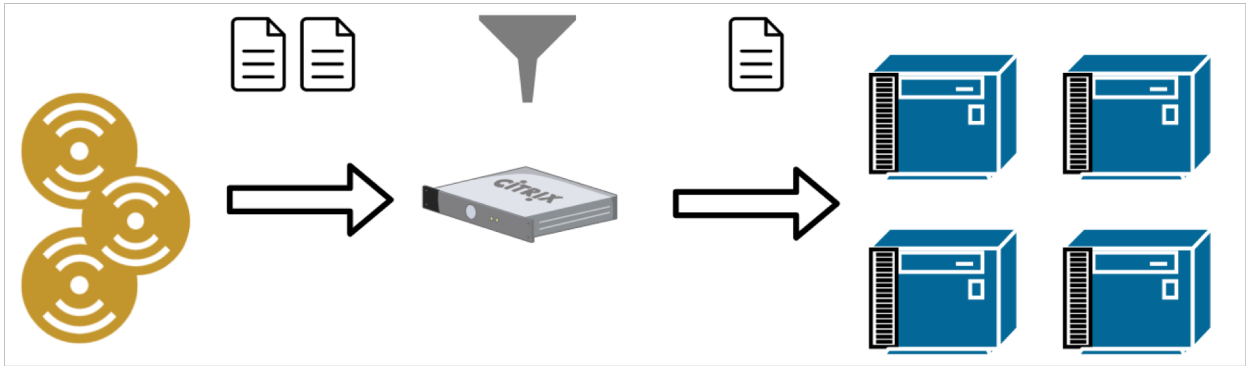


Quality of Service Requirements

- Distributed sensor systems increasingly deployed even in critical infrastructures → critical performance requirements, e.g.
  - end-to-end latencies
  - throughput
  - resource usage
  - reliability
  - availability

Edge and Fog Computing

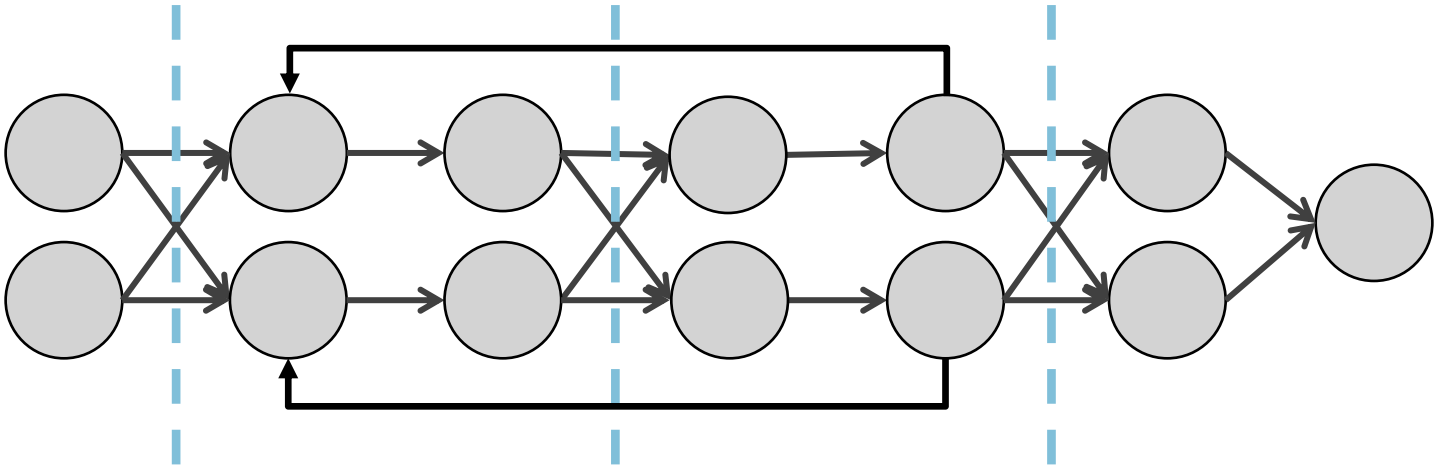
- From Edge to Fog to Cloud → execution of tasks on resources in-between sensors and Clouds



- Processing in increasingly heterogeneous, widely distributed computing environments

Distributed Dataflow Systems

- Popular tools for general-purpose scalable data processing, e.g. MapReduce, Spark, Flink, Beam
- Distributed execution of data-parallel operator graphs on shared-nothing commodity clusters



Processing of Continuous Streams

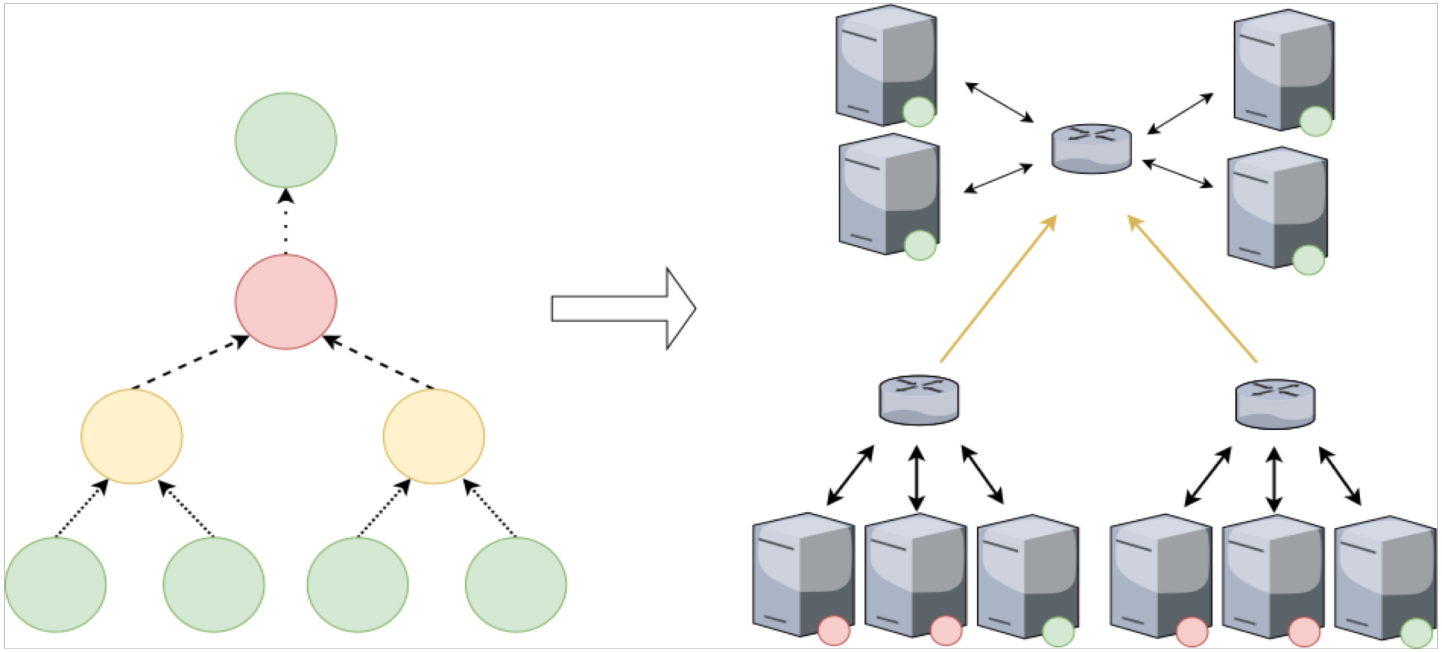
- Dedicated streaming systems (e.g. Storm), Microbatching (e.g. Spark Streaming), and unified dataflow engines (e.g. Flink and Beam) with
  - Low-latency processing of incoming data
  - Flexible windows for aggregation
  - Event time and out-of-order processing
  - Stream processing guarantees

Dataflow Task Scheduling

- At the moment, systems like Spark and Flink mostly expect to run on homogeneous clusters:
  - Same resource configuration per worker
  - Task scheduling without regard to topologies
  - Aggressive task chaining

- Yet, systems need to take the heterogeneity and geo-distribution of resources into account for IoT

Goal: Effective Task Placement



Taskgraphs:

- Resource requirements
- Task output

Infrastructure topologies:

- Latency
- Bandwidth
- Compute power

Comparison of Heuristic Methods

- Metric-based search methods:
  - Steepest descent
  - Tabu search
- Topology-based constructive methods:
  - Highest-level first (Edge first)
  - Top down (Cloud first)

Quality of Service Metrics

- Response Time (RT): time to transmit and process data items from source to sink tasks
- Resource Fitting (RF): assessment of the task placement by a distance function
- Bandwidth Congestion (BW): accumulated underprovisioning of the network

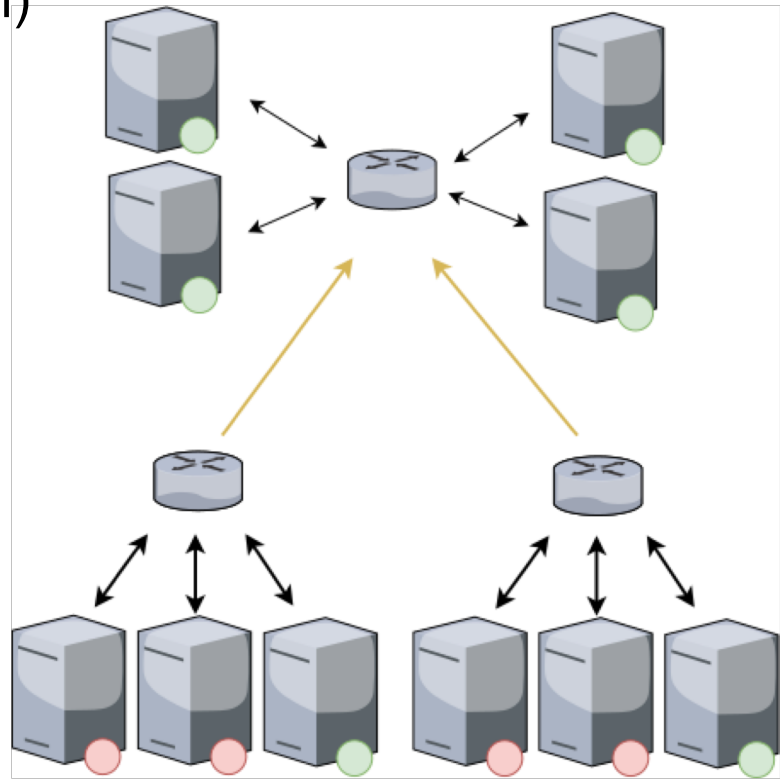
- Overall optimization function for placement P:

$$F(P) = \frac{RT(P) - RT_{\min}}{RT_{\max} - RT_{\min}} + \frac{RF(P) - RF_{\min}}{RF_{\max} - RF_{\min}} + \frac{BW(P) - BW_{\min}}{BW_{\max} - BW_{\min}}$$

Evaluation: Testbed

- Network emulation with GNS3 deployed on 9 machines (Intel Xeon E3-1230 V2 3.30GHz, 16 GB RAM, 1 Gbit/s Ethernet)

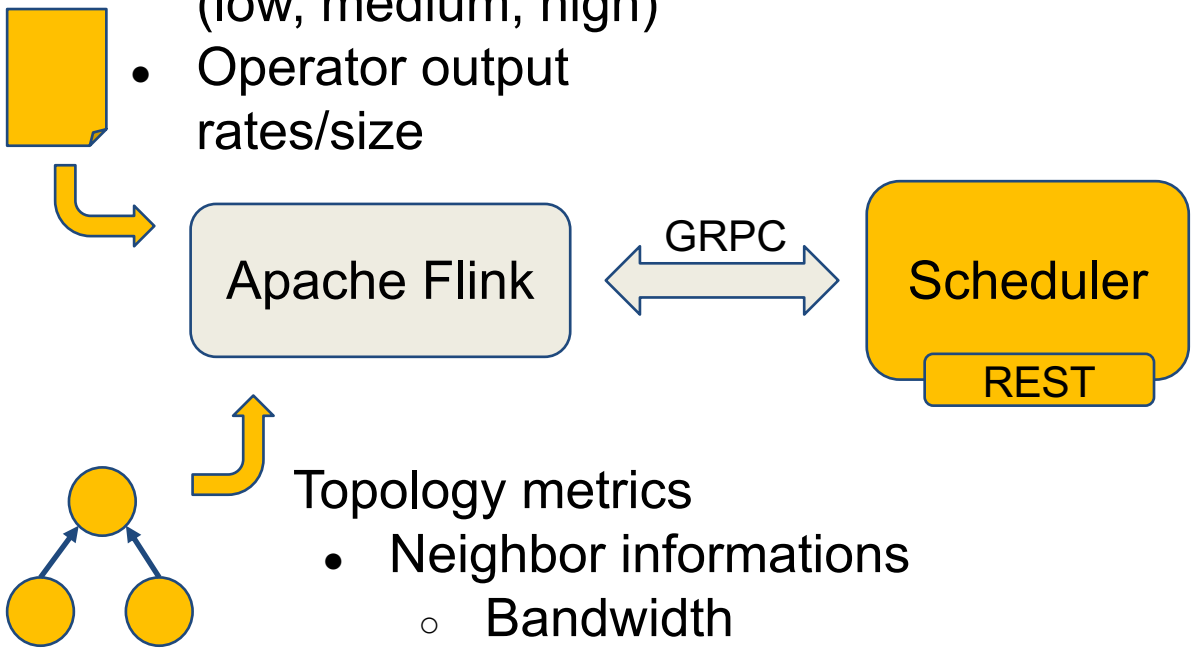
- Network and bandwidths (Open vSwitch)
  - Yellow (100 MB/s)
  - Black (no limitation)
- Latencies (NetEm)
  - Yellow (80ms)
  - Black (no limitation)
- CPU limitations (Docker)
  - RED - 1 core
  - Green - 4 cores



Evaluation: Prototype

Job metrics

- Resource requirements (low, medium, high)
- Operator output rates/size



Topology metrics

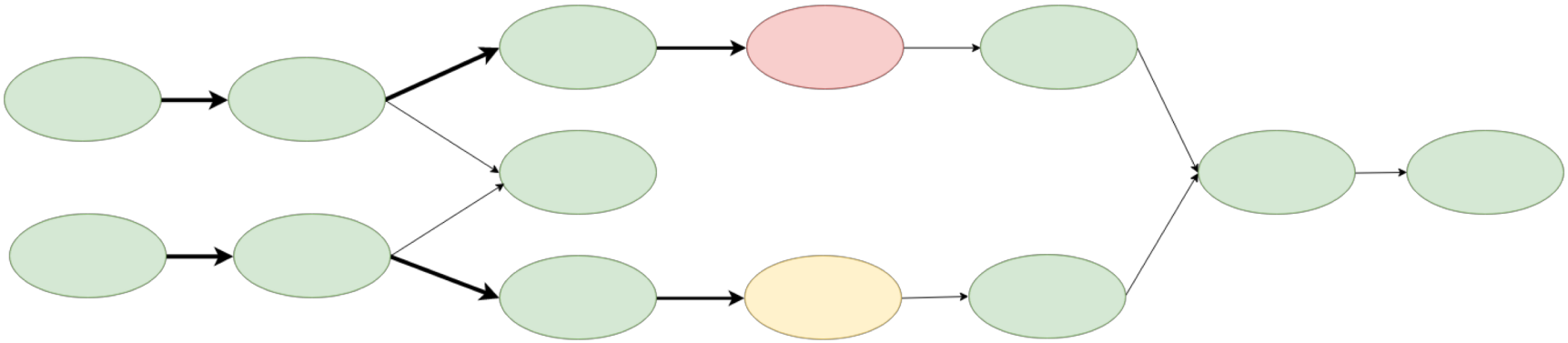
- Neighbor informations
  - Bandwidth
  - Latency
- Resource classification (low, medium, high)

Scheduler

- converts job/topology informations to graphs
- algorithms for topological- and metrics-based scheduling

Evaluation: Test Job

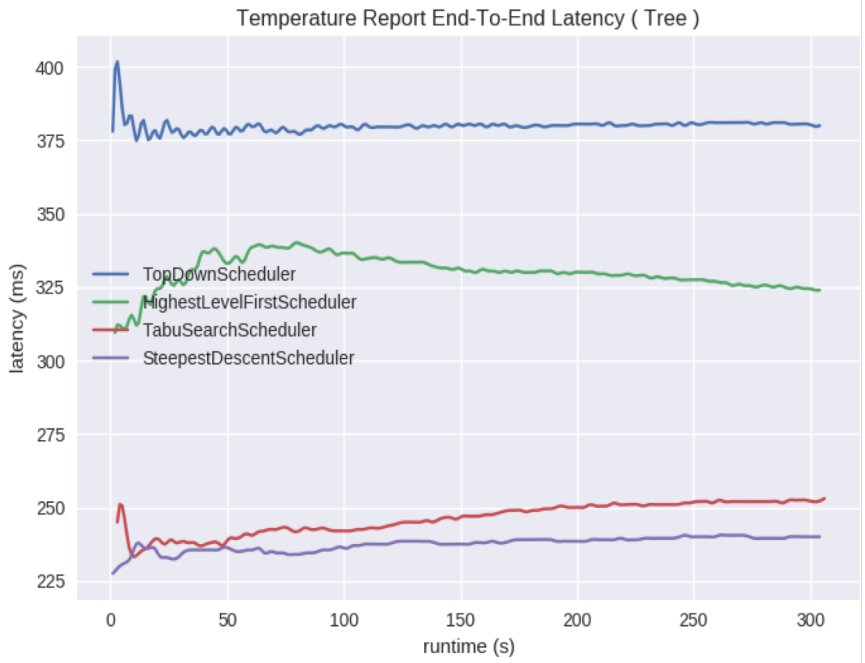
- Monitoring job for continuous temperature monitoring and fire detection

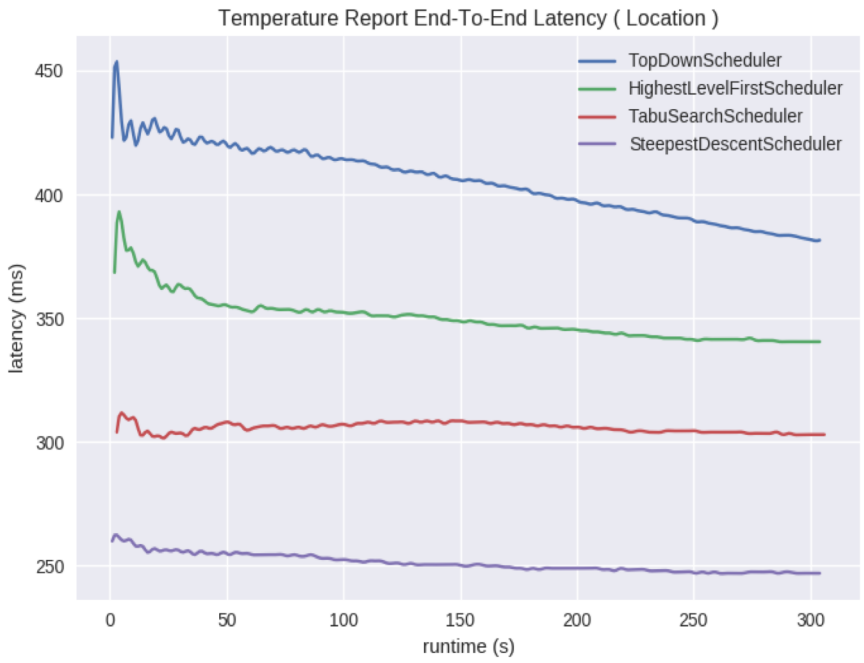


- Resource Classification: red -> high, yellow -> moderate, green -> low

Evaluation: Results

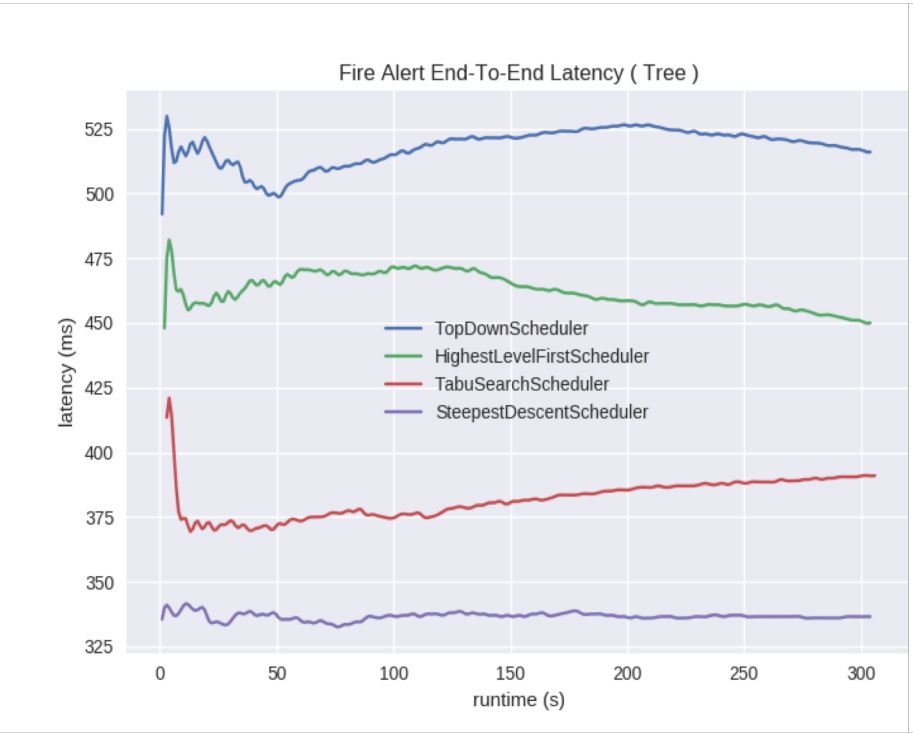
- Ingestion rate: 100 TreeReports and 25 LocationReports per second

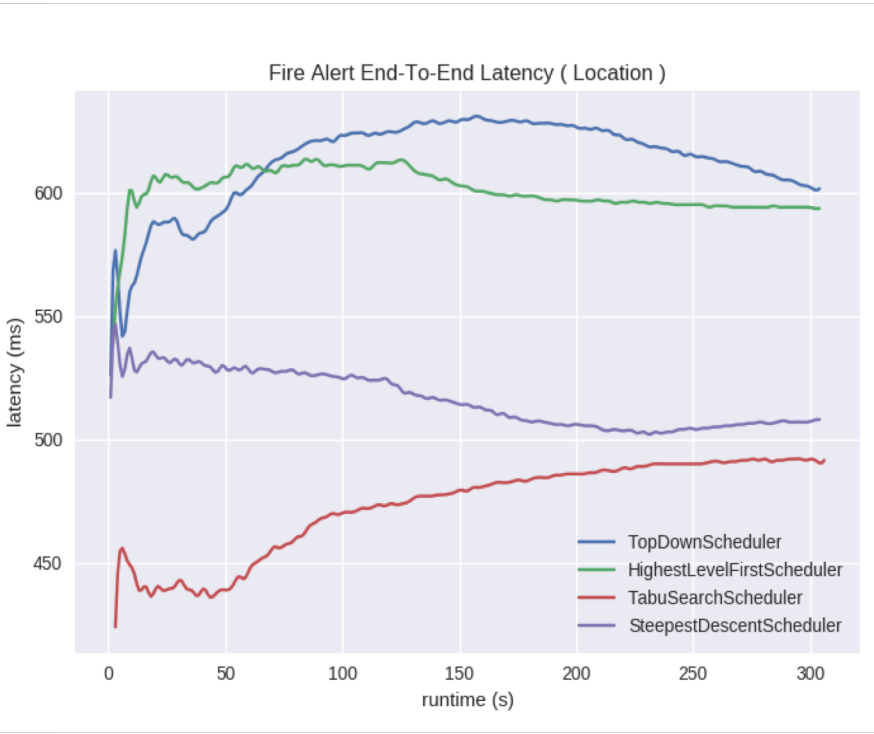




Evaluation: Results

- Ingestion rate: 100 TreeReports and 25 LocationReports per second





Conclusion

- Performance of the benchmark application varies drastically depending on the task placement
- Metrics-based methods yield up to 2x better schedules than the worst schedules in benchmark

- Yet:
  - Currently users need to provide additional inputs on their jobs
  - Environment and schedules are assumed static

Future Work

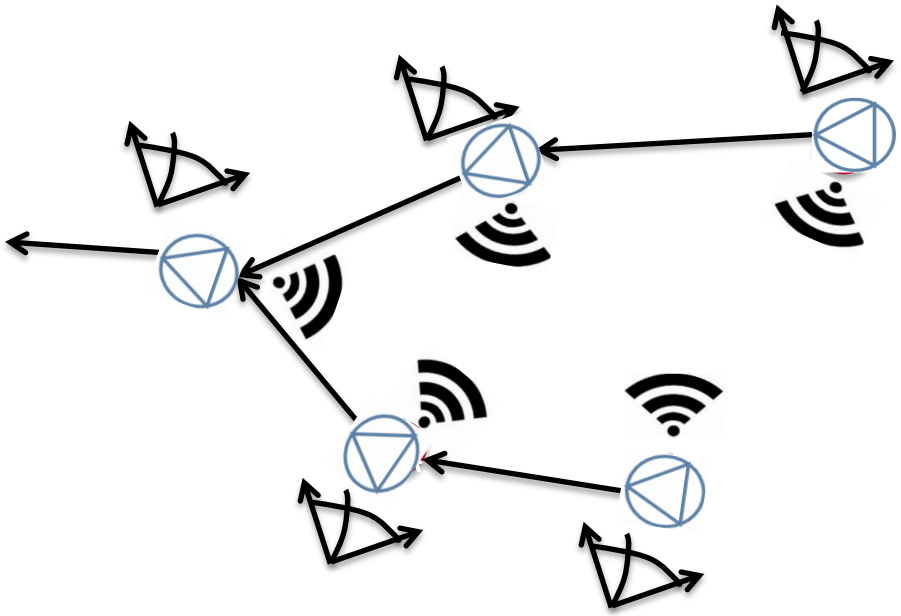
- Towards adaptive and fully transparent methods
  - Automatic inference of resource requirements
  - Continuous monitoring of job and topology metrics
  - Live adaptation of schedules (either re-scheduling or migration of tasks)

- More comprehensive empirical evaluation
  - More realistic test environments
  - More benchmarks

Overview

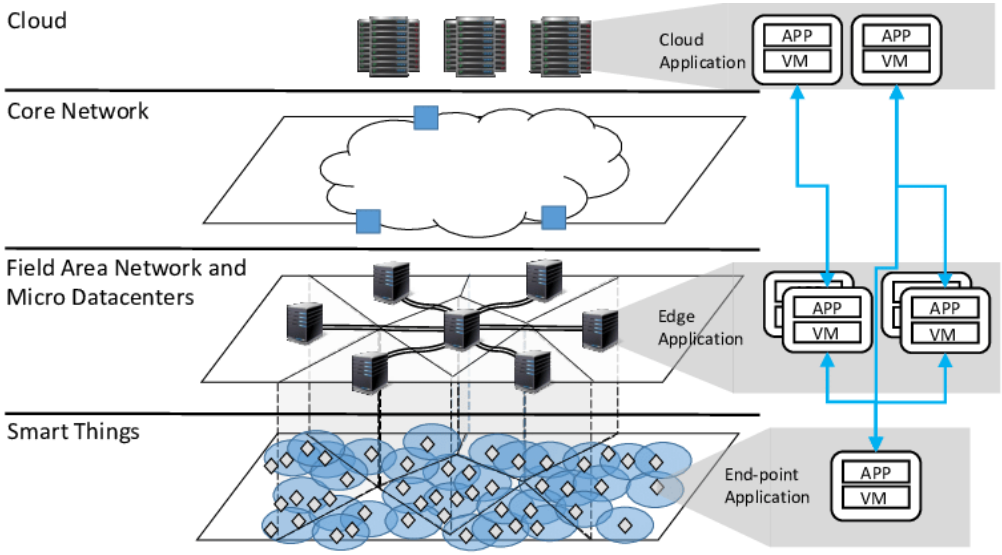
- Intro
- Federated, Hybrid and Multi-Clouds
  - Concepts
  - Policy enforcement
- Fog, Edge and IoT
  - Concepts
  - Placement in fog architectures
  - **Development practices: Testbed for fog computing**

Internet of Things Sensor Systems



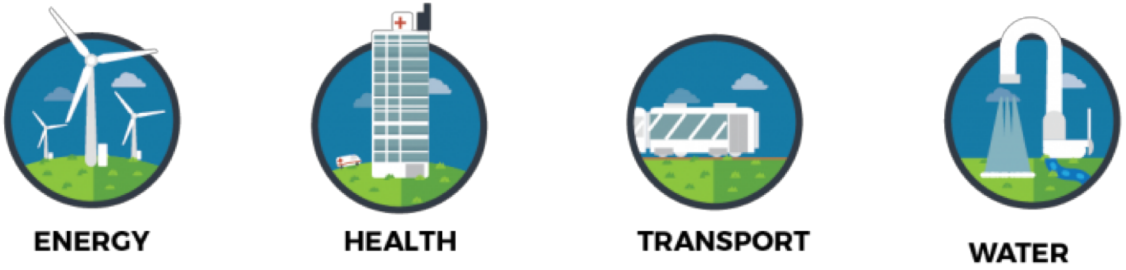
- Remote monitoring applications, e.g.
  - rule-based with thresholds, aggregate statistics,
  - clustering, classification, anomaly detection, or
  - input for simulations and system optimization

Edge/Fog Architectures of the IoT



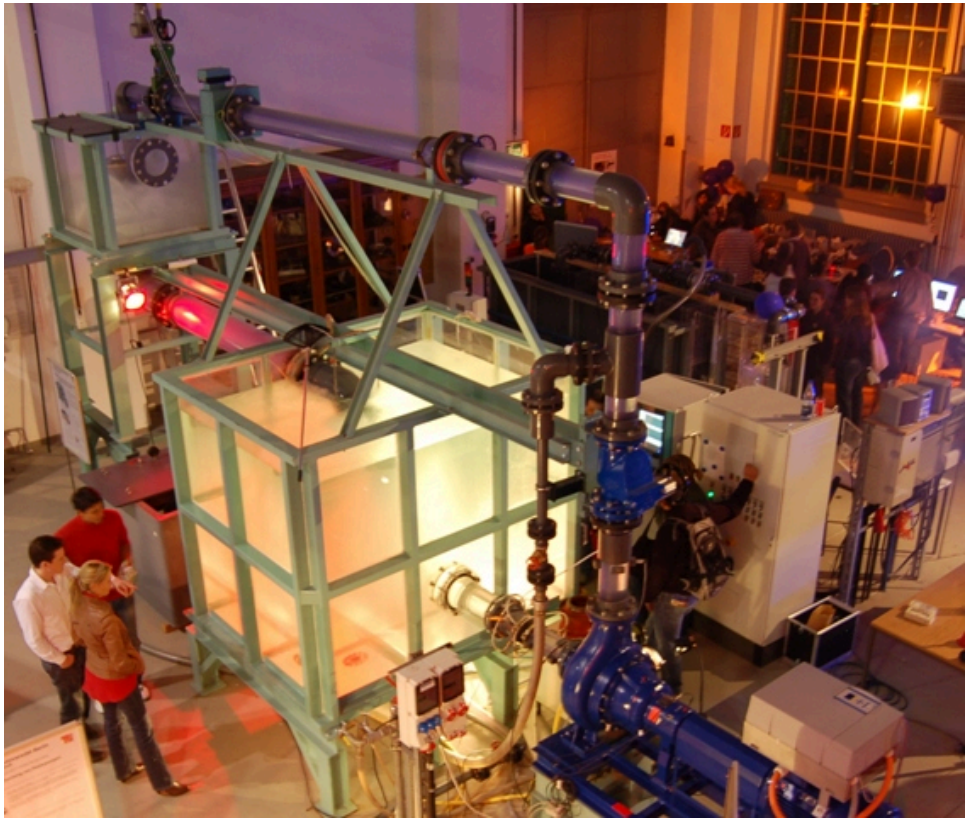
- Networks and nodes in IoT architectures:
  - Widely distributed and heterogenous
  - Dynamically changing systems and workloads (e.g. events, failures, weather, attacks)

Critical Service Constraints

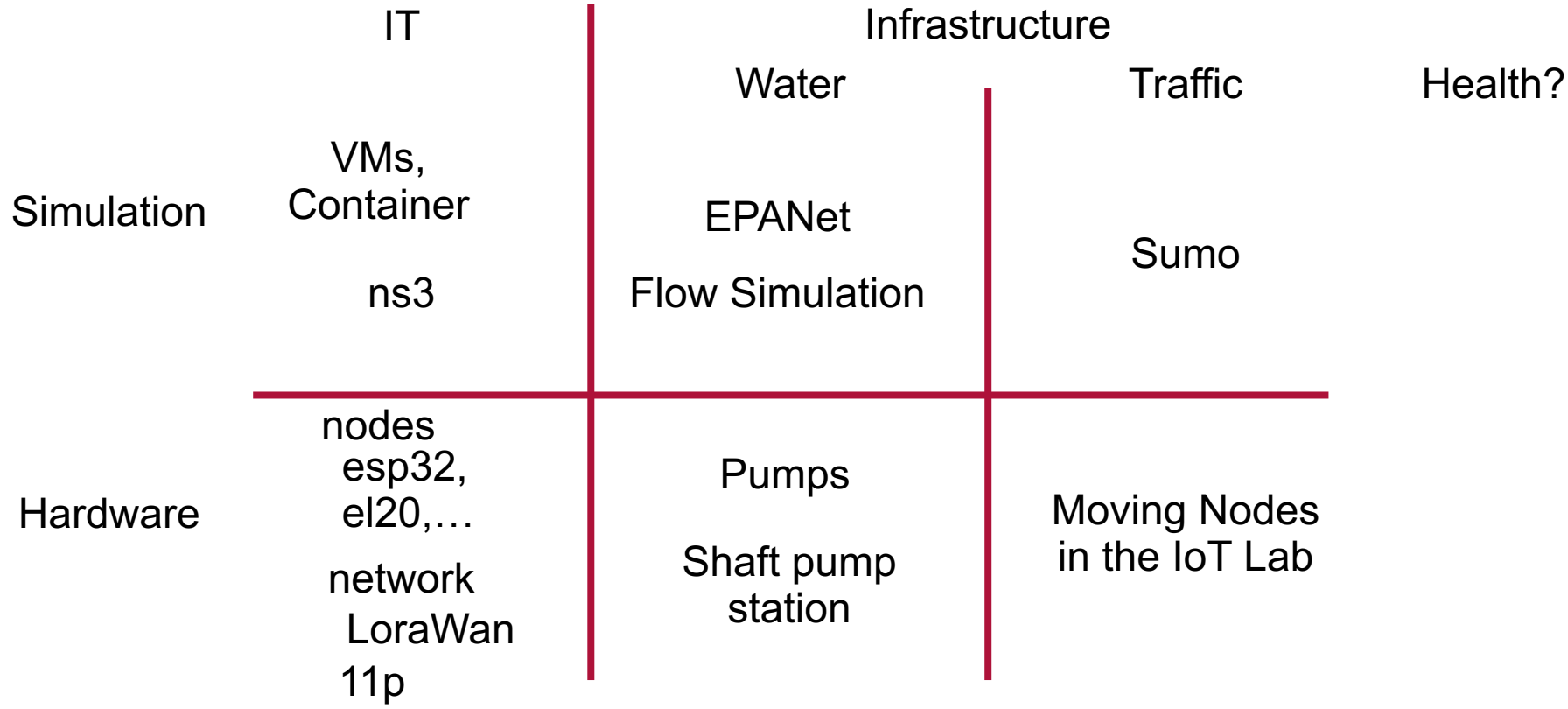


- Strong requirements in many critical IoT applications, e.g. in industrial and public infrastructures:
  - performance (latencies, resource usage)
  - reliability, availability, and security (exactly-once, max recovery times, uptime under stress/attack)

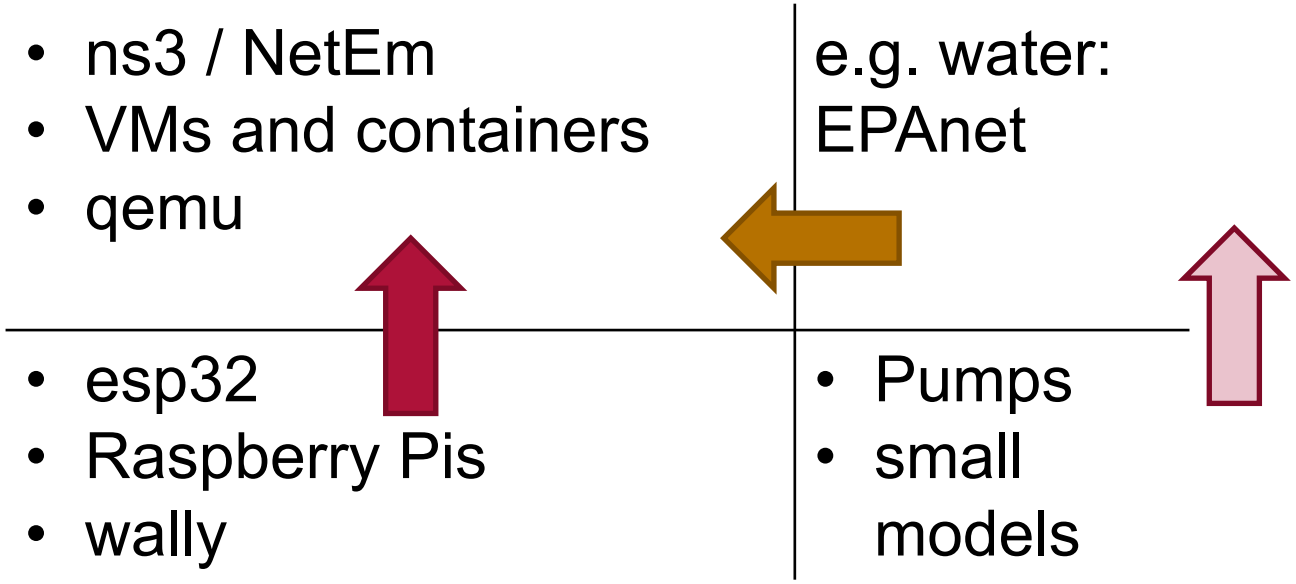
Hardware Testbeds



A Testbed for the Internet of Things



Integrated IoT Testbed (2/2)



Calibration (and continuous re-calibration)

Virtual sensors in simulations