Towards a Description of Elastic Cloud-native Applications for Transferable Multi-Cloud-Deployments

Peter-Christian Quint, Nane Kratzke

CoSA Center of Excellence

Lübeck University of Applied Sciences

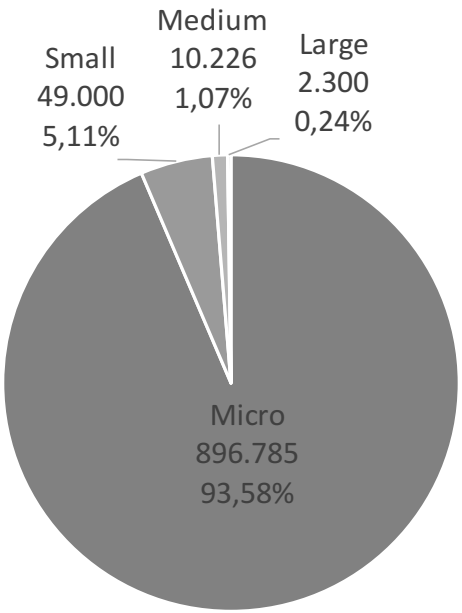
cloud TRANSIT

CoSA Center of Excellence

cloud TRANSIT

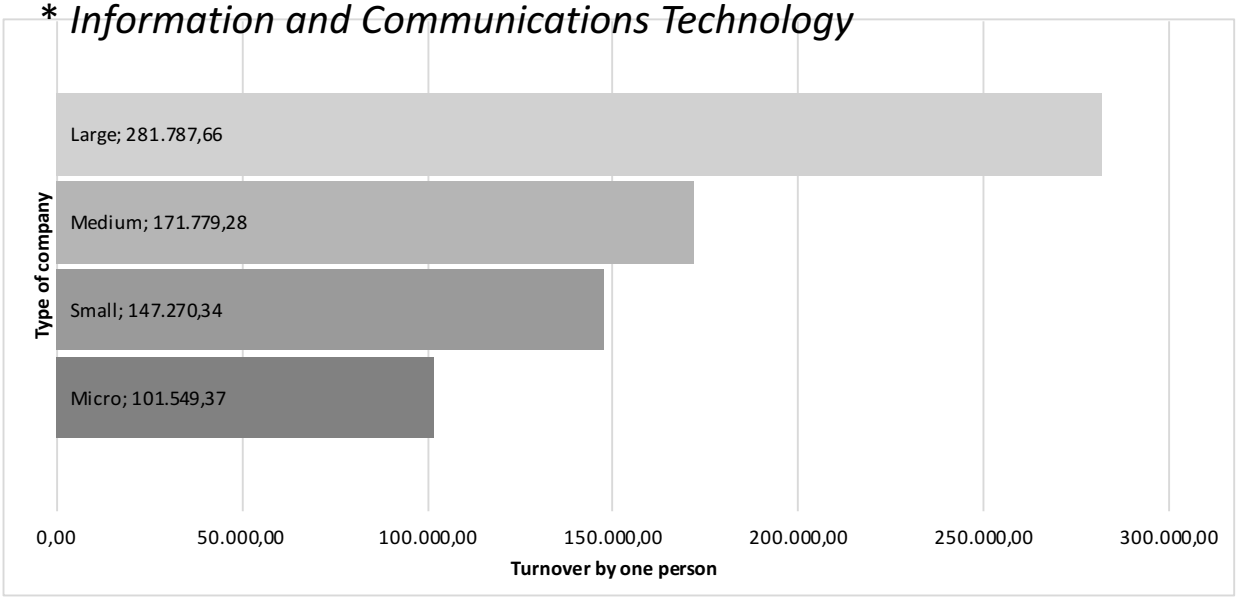
Small and medium-sized enterprises (SMEs)

According to EUSTAT..



93% of all ICT* companies are **micro companies** (< 10 persons)
But micro companies have only **1/3 of productivity**
Why?
E.g. due to natural limitations in their IT scalability

* *Information and Communications Technology*



- Personnel and financial restrictions
- Afraid of Vendor lock-in

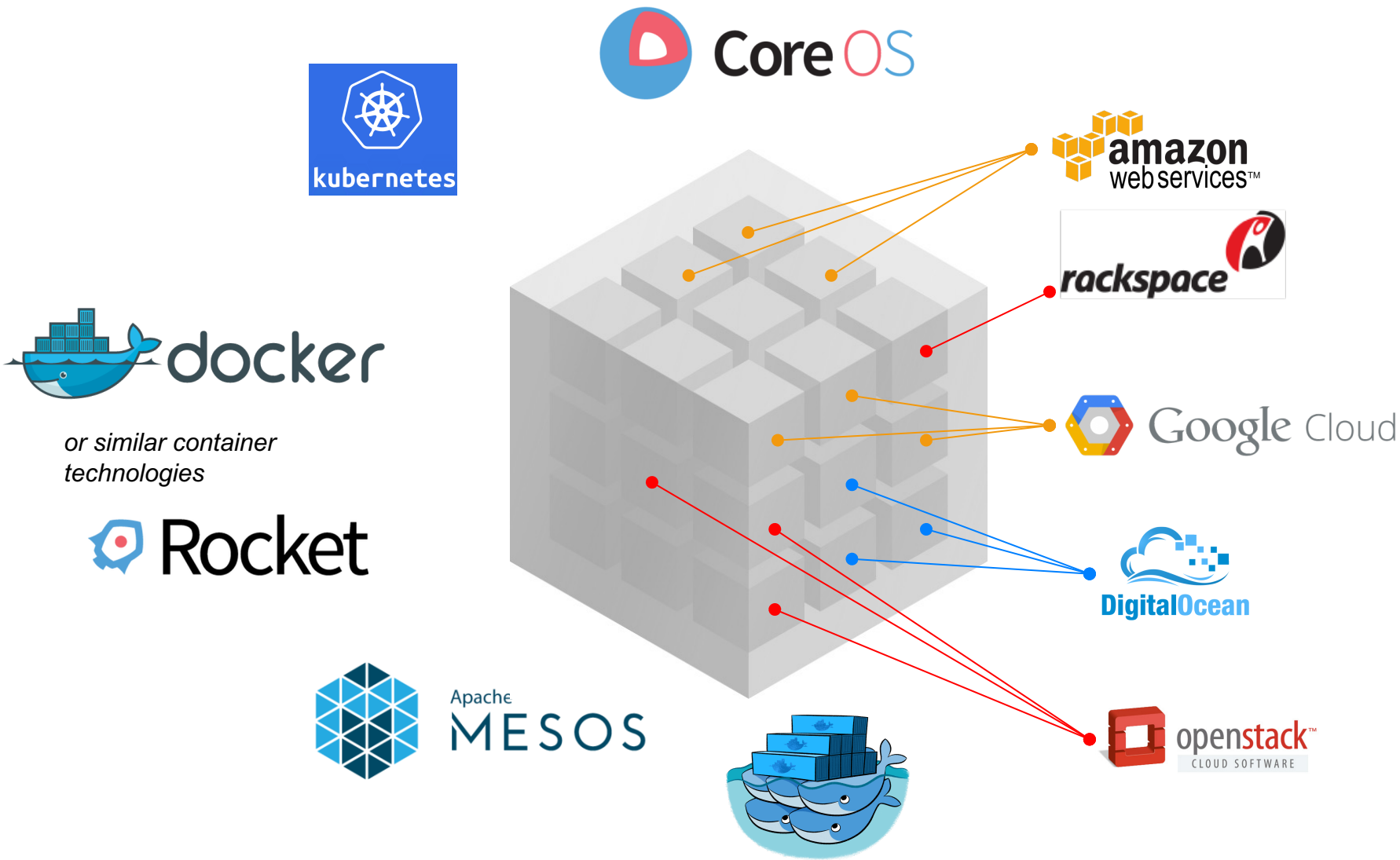
Research Focus

cloud TRANSIT

- The **aim** is to provide **methodologies** and **tools** to define secure, transferable and elastic services being deployable to any IaaS cloud infrastructure.
- Migration of these services from one private or public cloud infrastructure to another should be possible.
- The solution should be manageable by small and medium sized enterprises **(1-person IT staffs)**.

Quint, P.-C., & Kratzke, N. (2016). Overcome Vendor Lock-In by Integrating Already Available Container Technologies - Towards Transferability in Cloud Computing for SMEs. In Proceedings of CLOUD COMPUTING 2016 (7th. International Conference on Cloud Computing, GRIDS and Virtualization).

Container and Container Cluster



Research Focus

cloud TRANSIT

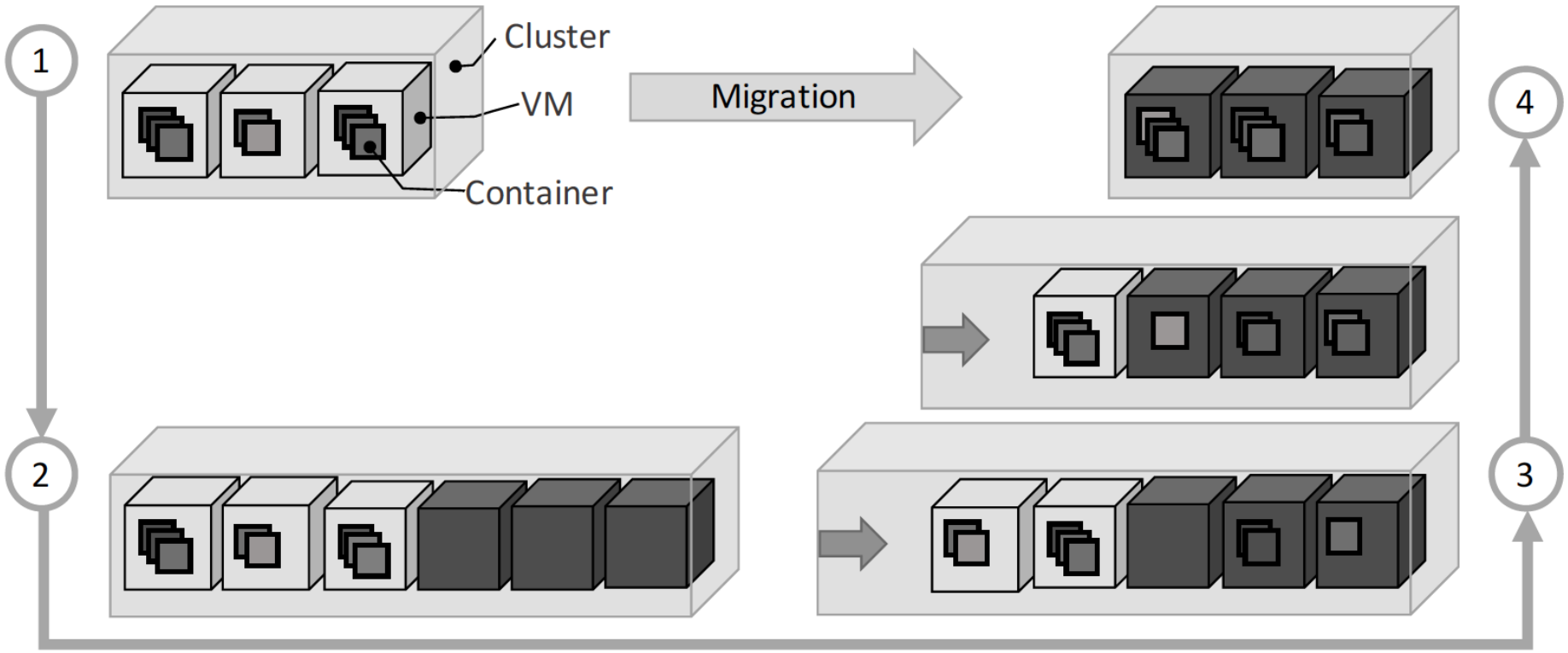


Fig. 1. Migrating IaaS platform. (1) Cluster hosted on VM by source provider A, (2) extending cluster with machines provided by target provider B, (3) terminating source provider VM, automatical detection and recreation of missing containers on target provider machines. (4) Migration complete.

Cloud-native Application

cloud TRANSIT

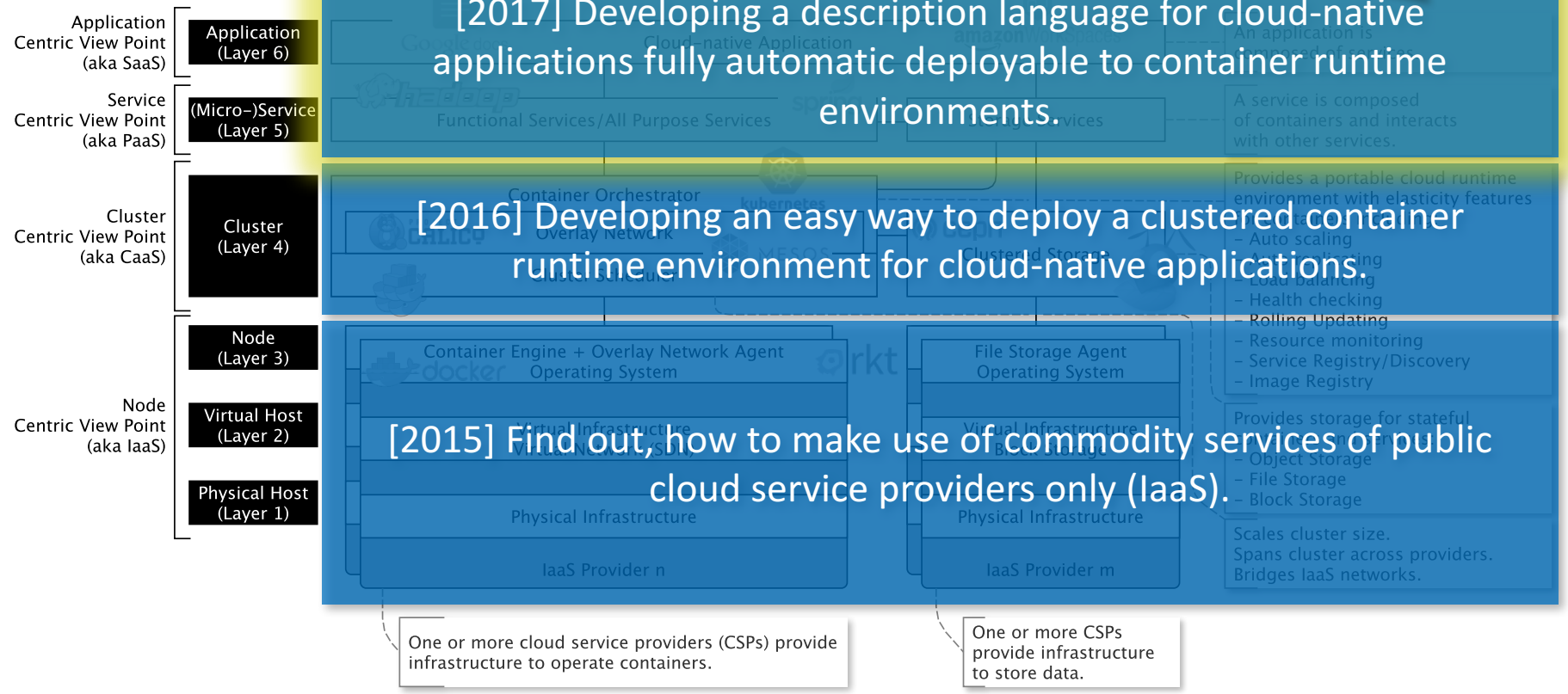
> A **cloud-native application** is a distributed, elastic and horizontal scalable system composed of (micro)services which isolates state in a minimum of stateful components. The application and each self-contained deployment unit of that application is designed according to cloud-focused design patterns and operated on a self-service elastic platform.

We call such a self-service elastic platform **Elastic Container Platform** (ECP)

Kratzke, N., & Quint, P.-C. (2017). Understanding Cloud-native Applications after 10 Years of Cloud Computing - A Systematic Mapping Study. *Journal of Systems and Software*, *126* (April).

Research focus Integrating three layers

Today's talk



Nane Kratzke and Rene Peinl. ClouNS - a Cloud-Native Application Reference Model for Enterprise Architects. In 2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW), pages 198–207, Vienna, sep 2016. IEEE

Split the migration problem into two independent engineering problems which are too often solved together.

1. The **infrastructure aware** deployment and operation of ECPs. These platforms can be deployed and operated in a way that they can be transferred across IaaS infrastructures of different private and public cloud service as (Kratzke, 2017) showed.
2. The **infrastructure agnostic** deployment of applications on top of these kind of transferable container platforms which is the focus of this paper.

Kratzke, N. (2017). Smuggling Multi-cloud Support into Cloud-native Applications using Elastic Container Platforms. In 8th. Int. Conf. on Cloud Computing and Service Sciences, Porto, Portugal.

Methodology

Three step DSL design methodology:

1. Analysis
2. Implementation
3. Use

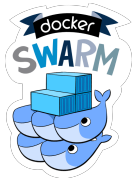
Van Deursen, A., Klint, P., and Visser, J. (2000). Domainspecific languages: An annotated bibliography. ACM Sigplan Notices, 35(6):26–36.

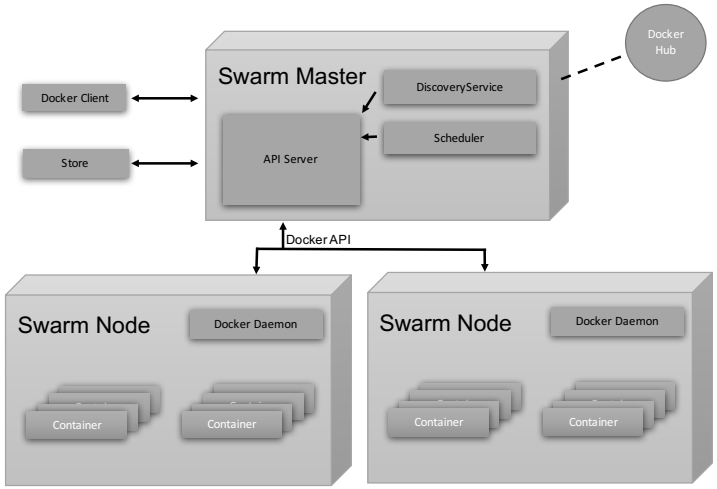
Mernik, M., Heering, J., and Sloane, A. M. (2005). When and how to develop domain-specific languages. ACM Computing Surveys, 37(4):316–344.

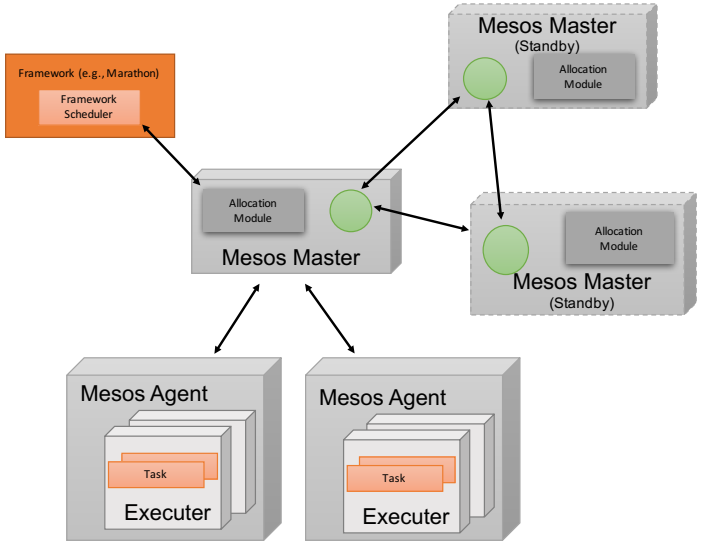
Strembeck, M. and Zdun, U. (2009). An approach for the systematic development of domain-specific languages. Software: Practice and Experience, 39(15):1253–1292.

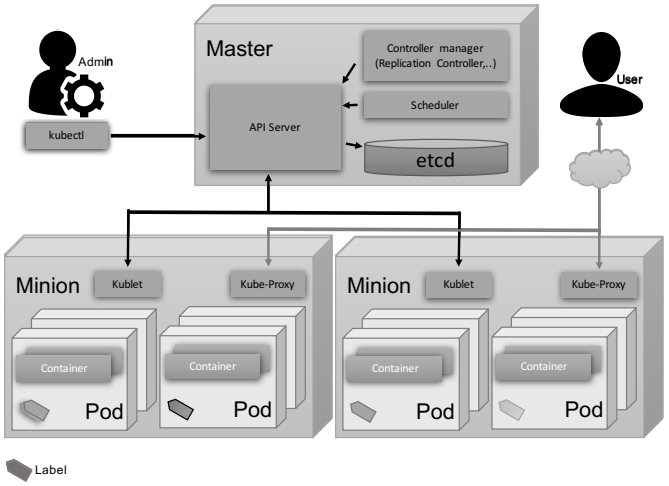
ECP Architectures

cloud TRANSIT









Components of ECP Architectures

cloud TRANSIT

Component	Mesos	Docker Swarm Mode	Kubernetes
Application Definition	Application Group	Compose	Service + Namespace Controller (Deployment, DaemonSet, Job, ...) All K8S concepts are described in yaml
Service Discovery	Mesos DNS	Service names Service links	KubeDNS (or replacment)
Deployment Unit	Binaries Pods (Marathon)	Container (Docker)	Pod (Docker, rkt)
Scheduling	Marathon Framework Constraints	Swarm scheduler Constraints	kube-scheduler Affinities + (Anti-)affinities
Load Balancer	Marathon-lb-autoscale	Ingress Load Balancing	Ingress controller Kube-proxy
Autoscaling	Marathon-autoscale	-	Horizontal pod autoscaling
Component Labeling	key/value	key/value	key/value

DSL Requirements for a ECP based CNA

cloud TRANSIT

R1 Containerized deployments.

- Microservice architecture, applications are split into small services which are designed to do one task well (Newman, 2015).
- Single services can be scaled independently to each other according to their individual use (Quint, 2017).
- ECPs such as Docker Swarm or Kubernetes are often exclusively designed to schedule and deploy container applications.

The DSL must be designed to describe a containerized deployments.

Sam Newman. Building Microservices. O'Reilly Media, 2015

Peter-christian Quint. Taming the Complexity of Elasticity , Scalability and Transferability in Cloud Computing - Cloud- Native Applications for SMEs Cloud-Native Applications for SMEs. 9(January):389–400, 2017.

DSL Requirements for a ECP based CNA

cloud TRANSIT

R2 Application Scaling

- Elasticity and scaling are one of the major advantages using cloud computing (Vaquero et al., 2011).
- Scalability enables quick scale up and scale down to rise the performance or free unused resources to reduce cost (Mao and Humphrey, 2011)

Scaling rules must describable

Vaquero, L. M., Rodero-Merino, L., and Buyya, R. (2011). Dynamically scaling applications in the cloud. ACM SIGCOMM Computer Communication Review, 41(1):45.

Mao, M. and Humphrey, M. (2011). Auto-scaling to min- imize cost and meet application deadlines in cloud workflows. Proceedings of 2011 International Con- ference for High Performance Computing, Network- ing, Storage and Analysis on - SC '11, page 1.

DSL Requirements for a ECP based CNA

cloud TRANSIT

R3 Compendiously

- To simplify operation the DSL should be pragmatic.
- Our approach is based on a separation between the description of the application and the elastic container platform.

- **The DSL must be designed to be lightweight and infrastructure-agnostic**.

DSL Requirements for a ECP based CNA

cloud TRANSIT

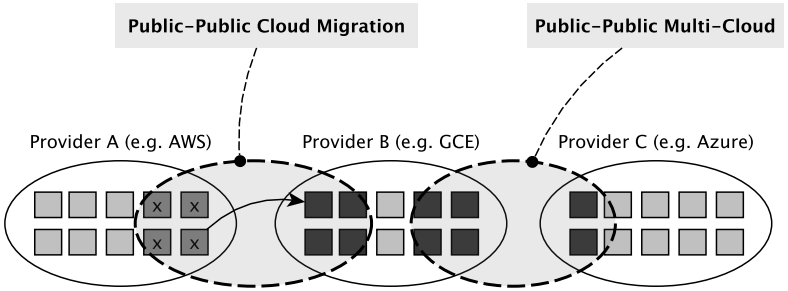
R4 Multi-Cloud-Support

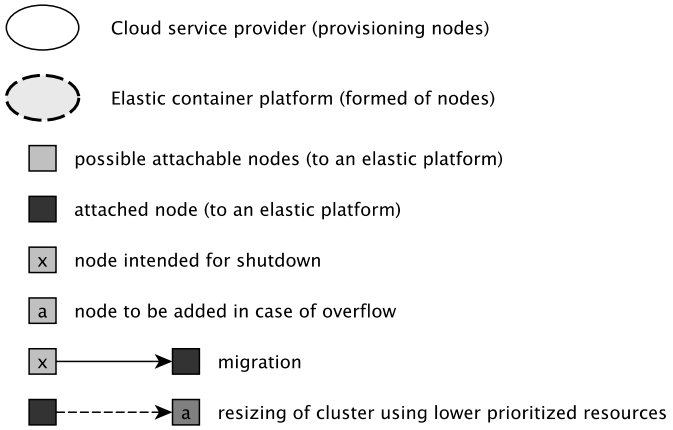
- Using multi-cloud- capable ECPs for deploying CNAs is a major requirement for our migration approach.
- Possibilities to optimize costs or improve quality of services.
- Adapt the cloud environment to new demands
- Multi-Cloud-Support also enables the use of Hybrid-Cloud-Infrastructures

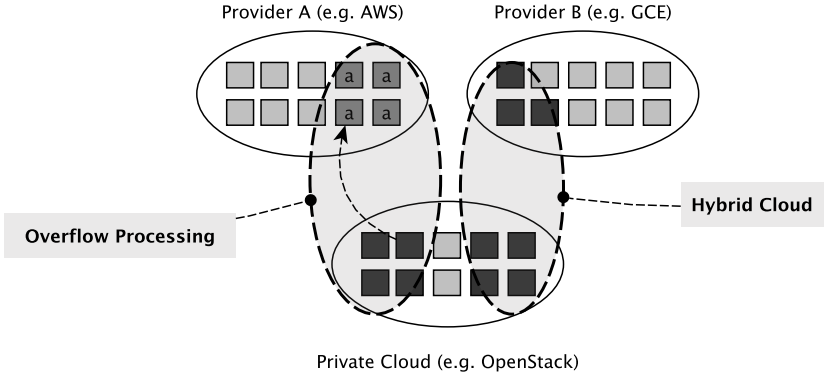
The DSL must designed to support multi-cloud operations

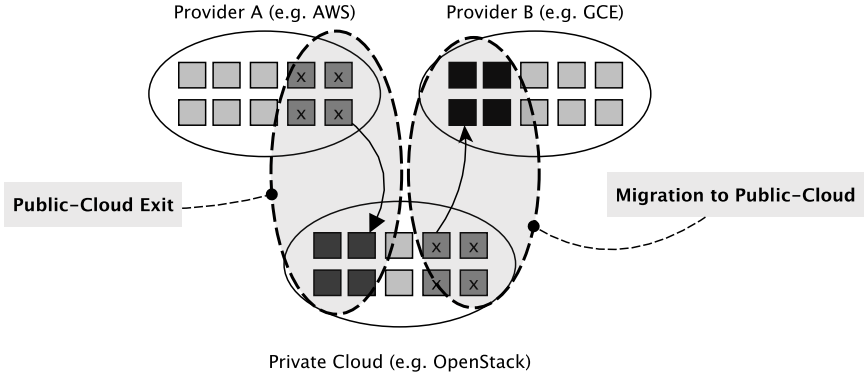
DSL Requirements for a ECP based CNA

R4 Multi-Cloud-Support.









DSL Requirements for a ECP based CNA

cloud TRANSIT

R5 Independence

- To avoid dependencies, the CNA should be deployable independent to a specific ECP and also to a specific IaaS provider

The DSL must be designed to be independent from a specific ECP or cloud infrastructure.

DSL Requirements for a ECP based CNA

cloud TRANSIT

R6 Adoption in run-time

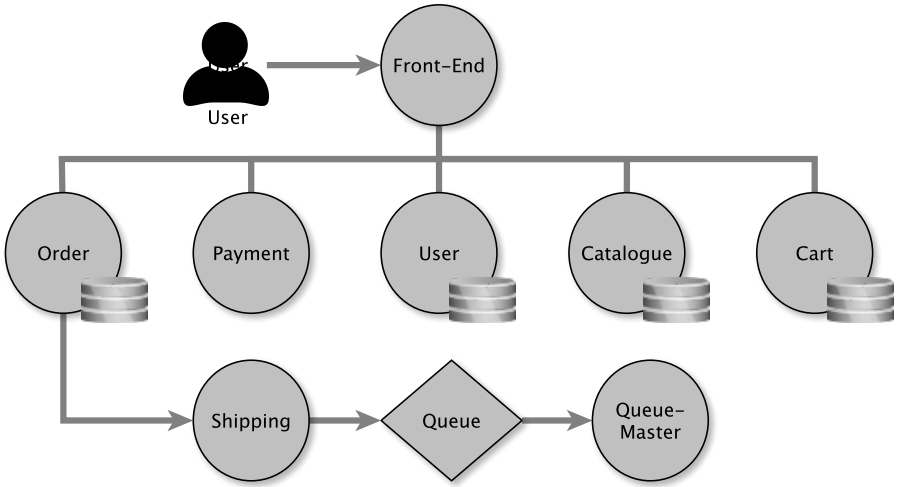
- The CNA must be deployable to a non-static runtime environment
- Description model must be adaptable to changes of the ECP runtime environment

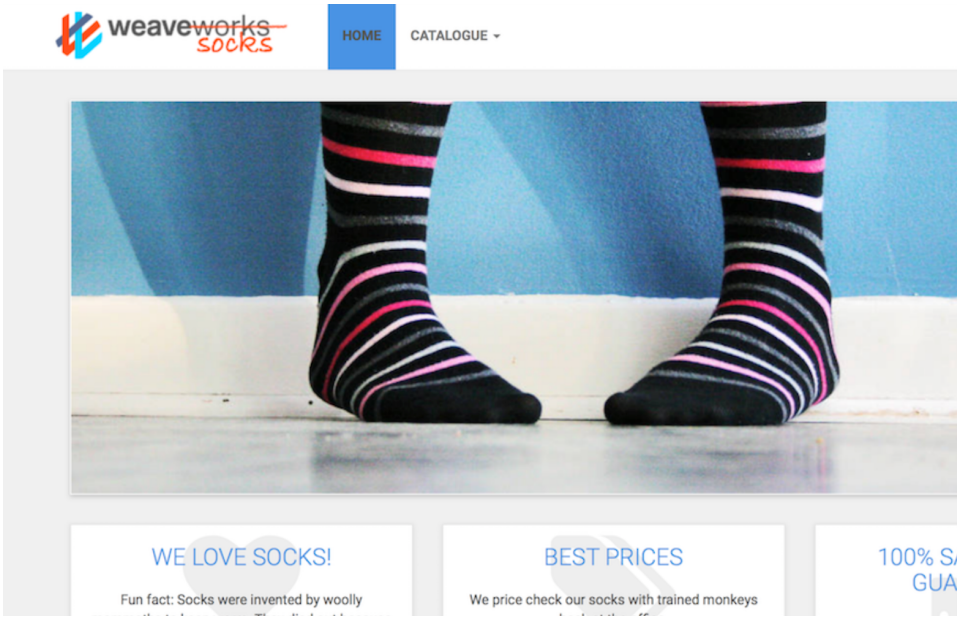
The DSL implementation must be able to support a non-static runtime environment.

Investigated DSL

- TOSCA (OASIS Topology and Orchestration Specification for Cloud Applications)
- CAML (Cloud Application Modeling Language)
- MULTICLAPP
- CloudML
- MODACloudsML (MOdel-Driven Approach for design and execution of applications on multiple Clouds)
- Docker Compose DSL
- Kubernetes DSL

Demo-Application: Weaveworks SockShop





Environment: Kubernetes on OpenStack Mitaka

SockShop: weave.works/sock-shop-microservices-demo-application/

Requirement Matching

cloud TRANSIT

	Container	Multi-Cloud	Scalability	Design in run-time	Independence	Compendiously	Implementation
CAML		+	+	+	+	-	R
CloudML	+	+	+	+	+	-	R
Docker Compose	+	+	+	+	-	+	P
Kubernetes	+	+	+	+	-	+	P
TOSCA	+	+	+	-	+	-	P+R
MODACloudML	+	+	+	+	+	-	R
MULTICLAPP		+	-	-	+		-

Outcome

No DSL fits all of our requirements.

TOSCA
- High tool-chain complexity
- Tendency to cover all layers of abstraction

Compose and Kubernetes
- Fulfill the most of our requirements
- Designed for a specific ECP (Docker Swarm and Kubernetes).

We identified the need for creating a new DSL
(at least for our research activities)

Acknowledgment

This research is funded by German Federal Ministry of Education and Research (Project Cloud TRANSIT, 13FH021PX4). The author thank the University of Lübeck (Institute of Telematics) and fat IT solution GmbH (Kiel) for their support of Cloud TRANSIT.

CoSA
Center of Excellence
Communications – **S**ystems – **A**pplications

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

fat IT SOLUTIONS

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung
Förderkennzeichen: **03FH021PX4**

FORSCHUNG AN FACHHOCHSCHULEN

All used images are CC0 Public Domain / https://pixabay.com/

About me

Dipl.-Inform.

Peter-Christian Quint

Mail: peter-christian.quint@fh-luebeck.de

Web: http://www.pcquint.de/

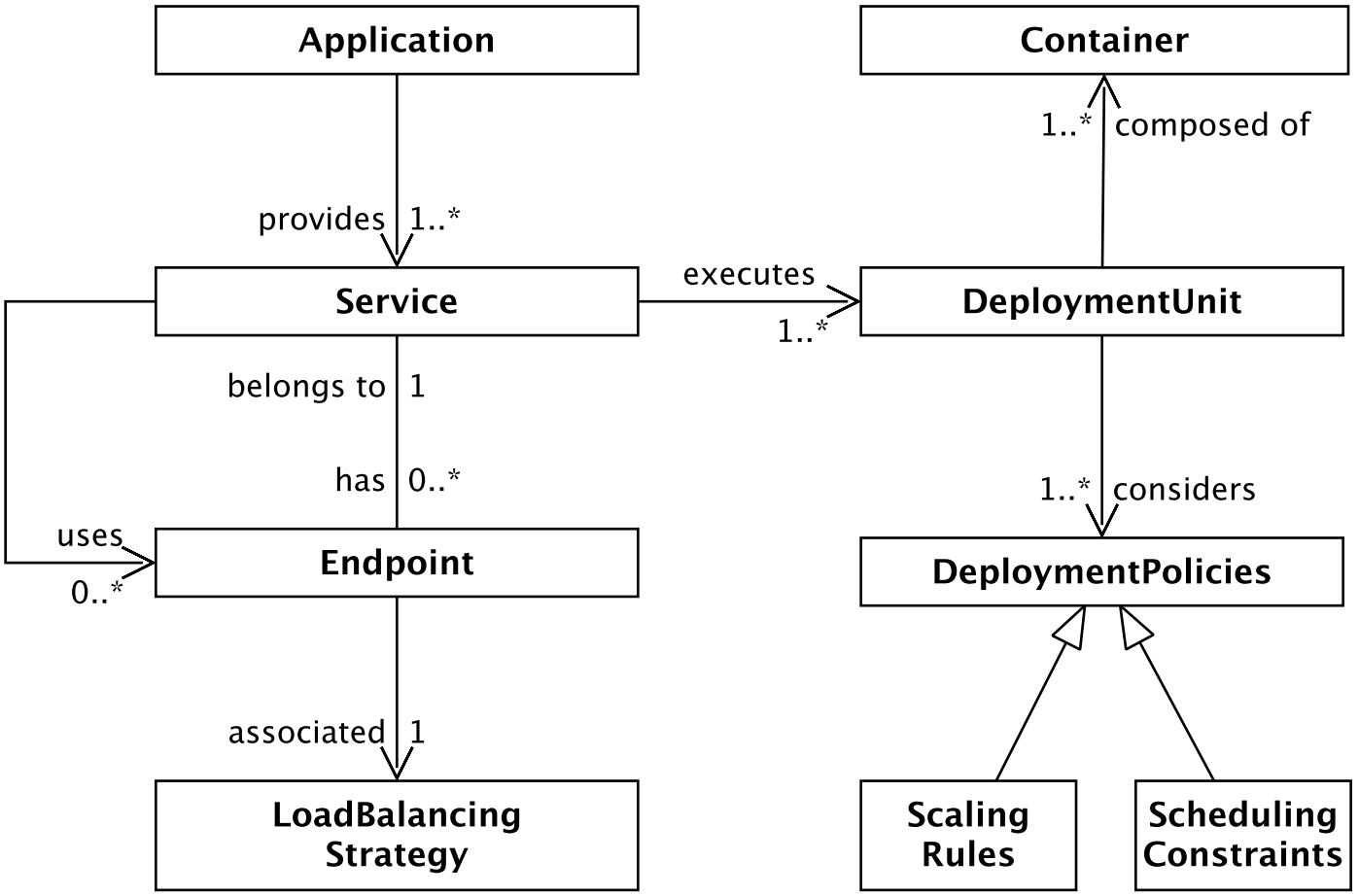
CoSA: http://cosa.fh-luebeck.de/en/contact/people/quint

ResearchGate: https://www.researchgate.net/profile/Peter_Christian_Quint2

XING: https://www.xing.com/profile/PeterChristian_Quint

Backup Slides

Core Language Model

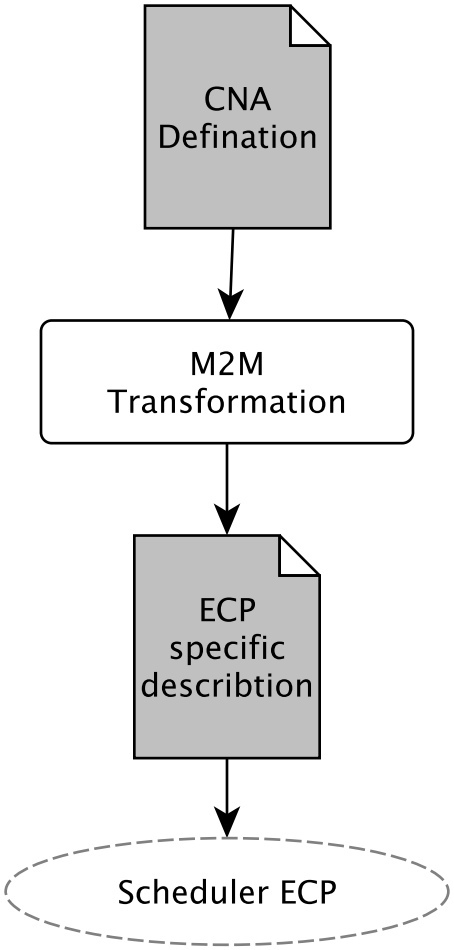


Example

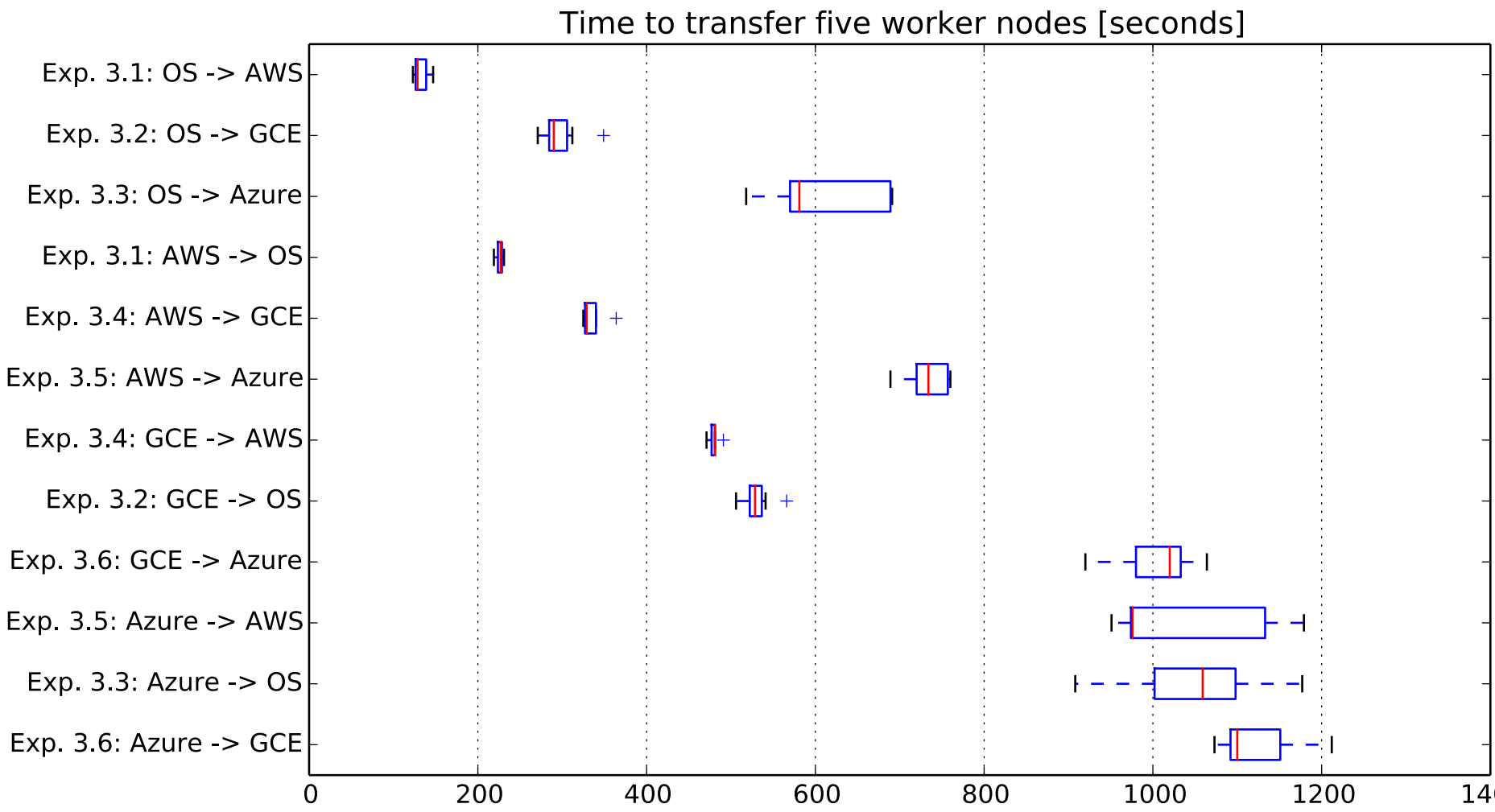
Listing 1: The payment service of the SockShop reference application expressed in the proposed DSL

```
1  DeploymentPolicies policies = new DeploymentPolicies.Builder()
2          .rule(DeploymentPolicies.Type.NUMBER, 3)
3          .rule(DeploymentPolicies.Type.SELECTOR, "openStack.dc1")
4          .build();
5  Container paymentContainer = new Container.Builder("payment")
6          .image("weaveworksdemos/payment:0.4.3")
7          .port(new Port.Builder().containerPort(80).build())
8          .build();
9  DeploymentUnit deploymentUnit = new DeploymentUnit.Builder("payment")
10         .container(paymentContaienr)
11         .tag("app", "nginx")
12         .deploymentPolicy(policies)
13         .build();
14 Service service = new Service.Builder("payment")
15         .deploymentUnit(deploymentUnit)
16         .port(new Port.Builder("http")
17         .protocol(Port.Protocol.TCP).containerPort(80).targetPort(80).build())
18         .build();
19
20 new Generator.Builder().targetECP(Generator.ECP_TYPES.KUBERNETES)
21         .deyploment(service)
22         .build()
23         .write(new File("/path/to/folder"));
```

M2M Transformation



Cloud Migration



Lessons Learned

cloud TRANSIT

Different quality and topicality of DSL implementations

The practical testing of DSLs is made more difficult because of a partial lack of usable implementations. E.g., the prototype implementation of CloudML seems not to be maintained any more and only useable by installing deprecated software (like Java 7) or by programming effort. Also Alien4Cloud, a TOSCA implementation which is still under active maintenance, was complex to install on our infrastructure. But there exist also implementations like Cloudify which are so highly developed that they can also be used for roductive environments. Hence, it can be complex (and time consuming) to try CNA describing in practice. When choosing a description solution for CNAs, the development level should be considered next to the functional scope, too.

Lessons Learned – Adaption in Runtime

cloud TRANSIT

Docker Compose and frameworks like Cloudify, CloudFoundry and Scalar are able to provide multi-cloud deployments with self-adapting features.

However, these approaches are not able to adapt the run-time environment of an already running system.

For a IaaS provider migration, the deployment has to terminate and restart with the adapted model. Correspondingly, a transfer of CNAs between different cloud providers is not possible during runtime.

Lessons Learned – Simplicity

cloud TRANSIT

”Everything should be as simple as possible, but not simpler.”
Albert Einstein

Describing, deploying and managing multi-cluster deployments should be possible in a obvious manner. Tools like Docker Compose meet the first part of the quotation. However, solutions like these don’t have the necessary scope to fulfill all of our requirements. On contrary, our future approaches for finding a solution for describing ECP based CNA according to our requirements will follow the principle of simplicity.

Similarity values of AWS and GCE virtual machines types

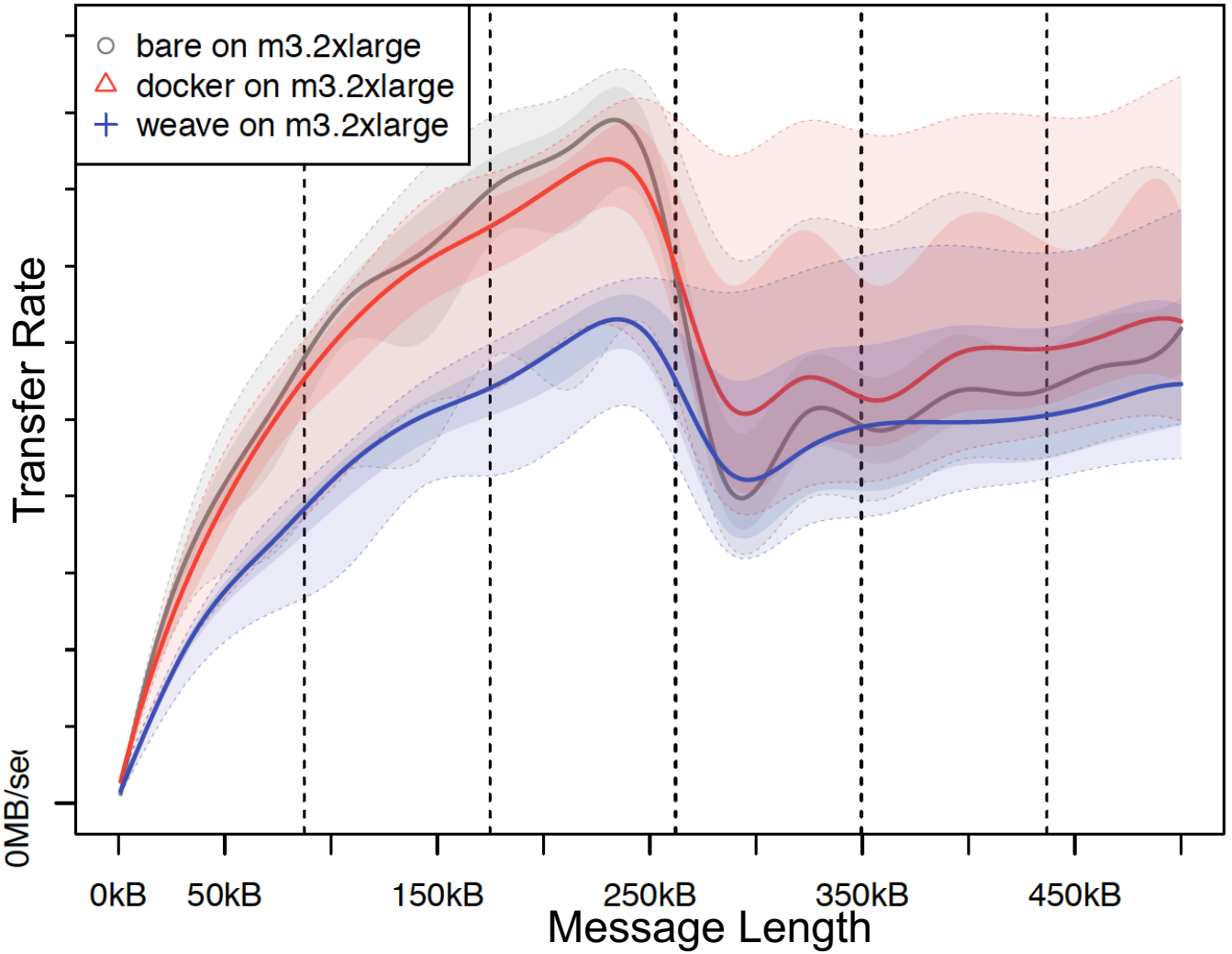
cloud TRANSIT

Similarities	n1-highmem-32	n1-standard-32	n1-highcpu-32	n1-highmem-16	n1-standard-16	n1-highcpu-16	n1-highmem-8	n1-standard-8	n1-highcpu-8	n1-highmem-4	n1-standard-4	n1-highcpu-4	n1-highmem-2	n1-standard-2	n1-highcpu-2	n1-standard-1	g1-small	f1-micro
m4.10xlarge	0.67	0.70	0.68	0.56	0.50	0.52	0.51	0.46	0.48	0.40	0.41	0.43	0.36	0.37	0.40	0.35	0.30	0.13
c4.8xlarge	0.66	0.66	0.70	0.59	0.65	0.54	0.55	0.47	0.50	0.41	0.43	0.46	0.37	0.39	0.43	0.38	0.32	0.14
c3.8xlarge	0.81	0.82	0.81	0.56	0.64	0.51	0.52	0.45	0.47	0.39	0.40	0.42	0.34	0.35	0.39	0.33	0.27	0.13
i2.4xlarge	0.61	0.57	0.60	0.84	0.77	0.80	0.66	0.60	0.64	0.52	0.54	0.59	0.46	0.48	0.53	0.47	0.40	0.22
r3.4xlarge	0.60	0.57	0.60	0.83	0.76	0.79	0.65	0.59	0.62	0.50	0.52	0.56	0.44	0.45	0.51	0.44	0.37	0.21
m4.4xlarge	0.61	0.58	0.61	0.84	0.77	0.80	0.66	0.60	0.64	0.51	0.54	0.58	0.45	0.47	0.52	0.46	0.39	0.21
c3.4xlarge	0.60	0.57	0.60	0.83	0.76	0.79	0.65	0.72	0.62	0.50	0.52	0.56	0.43	0.45	0.50	0.44	0.37	0.21
c4.4xlarge	0.59	0.55	0.58	0.82	0.75	0.79	0.64	0.72	0.62	0.53	0.56	0.60	0.49	0.51	0.56	0.50	0.44	0.25
i2.2xlarge	0.52	0.49	0.51	0.62	0.63	0.65	0.79	0.81	0.83	0.60	0.62	0.63	0.52	0.53	0.54	0.51	0.43	0.27
r3.2xlarge	0.52	0.49	0.51	0.62	0.63	0.65	0.79	0.81	0.83	0.60	0.62	0.63	0.52	0.53	0.54	0.51	0.43	0.27
m4.2xlarge	0.46	0.43	0.45	0.55	0.57	0.60	0.74	0.77	0.78	0.57	0.60	0.60	0.49	0.51	0.50	0.53	0.54	0.37
m3.2xlarge	0.52	0.49	0.50	0.61	0.63	0.66	0.79	0.96	0.83	0.60	0.63	0.63	0.52	0.54	0.54	0.52	0.44	0.27
c3.2xlarge	0.52	0.50	0.51	0.62	0.64	0.80	0.79	0.82	0.83	0.60	0.76	0.62	0.52	0.53	0.53	0.51	0.44	0.27
c4.2xlarge	0.53	0.51	0.52	0.62	0.64	0.80	0.78	0.81	0.82	0.58	0.74	0.60	0.50	0.52	0.52	0.50	0.42	0.26
i2.xlarge	0.45	0.42	0.43	0.54	0.59	0.59	0.58	0.78	0.62	0.81	0.82	0.80	0.60	0.60	0.58	0.57	0.48	0.32
r3.xlarge	0.46	0.42	0.43	0.54	0.59	0.60	0.58	0.79	0.63	0.82	0.83	0.81	0.62	0.62	0.60	0.58	0.50	0.33
m4.xlarge	0.47	0.44	0.45	0.56	0.61	0.61	0.60	0.66	0.64	0.83	0.85	0.82	0.63	0.63	0.60	0.60	0.51	0.34
m3.xlarge	0.47	0.43	0.45	0.56	0.58	0.74	0.60	0.63	0.64	0.80	0.96	0.82	0.59	0.61	0.61	0.58	0.49	0.31
c3.xlarge	0.46	0.42	0.43	0.54	0.60	0.59	0.58	0.65	0.77	0.82	0.83	0.79	0.61	0.74	0.57	0.56	0.48	0.32
c4.xlarge	0.49	0.46	0.47	0.57	0.62	0.62	0.61	0.67	0.79	0.81	0.83	0.80	0.59	0.74	0.57	0.57	0.48	0.32
r3.large	0.39	0.36	0.37	0.45	0.54	0.65	0.48	0.56	0.52	0.64	0.76	0.58	0.83	0.81	0.77	0.65	0.57	0.40
m3.large	0.41	0.38	0.39	0.48	0.54	0.53	0.51	0.58	0.69	0.63	0.64	0.61	0.84	0.98	0.80	0.67	0.58	0.39
c3.large	0.39	0.37	0.37	0.46	0.54	0.51	0.49	0.57	0.52	0.64	0.63	0.58	0.84	0.81	0.77	0.79	0.56	0.39
c4.large	0.40	0.37	0.37	0.46	0.55	0.51	0.50	0.58	0.53	0.65	0.63	0.59	0.81	0.79	0.75	0.78	0.56	0.40
t2.medium	0.43	0.40	0.40	0.49	0.56	0.52	0.53	0.59	0.55	0.66	0.62	0.59	0.74	0.70	0.68	0.69	0.48	0.31
m3.medium	0.31	0.30	0.30	0.34	0.42	0.39	0.36	0.43	0.40	0.50	0.47	0.44	0.57	0.55	0.51	0.86	0.77	0.57
t2.small	0.38	0.36	0.37	0.43	0.50	0.47	0.45	0.52	0.49	0.61	0.58	0.54	0.66	0.63	0.74	0.78	0.87	0.52
t2.micro	0.31	0.30	0.29	0.34	0.37	0.36	0.37	0.40	0.38	0.47	0.46	0.45	0.53	0.52	0.50	0.68	0.63	0.46

cloud TRANSIT

N. Kratzke and P.-C. Quint, “About automatic benchmarking of IaaS cloud service providers for a world of container clusters,” Journal of Cloud Computing Research, vol. 1, no. 1, 2015, pp. 16–34.

Absolute performance impact on transfer rates



N. Kratzke and P.-C. Quint, How to Operate Container Clusters more Efficiently? Some Insights Concerning Containers, Software-Defined-Networks, and their sometimes Counterintuitive Impact on Network Performance