university of groningen

Cost-efficiency in software (re-)architecting

Vasilios Andrikopoulos

Software Engineering and Architecture (SEARCH) Group
Bernoulli Institute for Mathematics, Computer Science
& Artificial Intelligence

SEN Symposium 2020
Amsterdam, January 31 2020

Bio/Research interests

Tilburg University

Service Engineering

- Services Evolution
- Compatibility of Interfaces
- API Management

Universität Stuttgart

Cloud Computing

- Application Migration
- Middleware Cloudification
- Cloud-based Application Engineering
- Cloud Economics

university of groningen

Engineering Large Scale Distributed Systems (of Systems)

- Architecting
- Cost management

Take-away messages (preview)

1. There is a relation between software cost and architecture

2. Architecting for cost efficiency is not only non-trivial…

3. …but requires both design- and run-time aspects

Cost?

> Software development takes time and money. When commissioning a building project, you expect a reliable estimate of the cost and development time up front. Getting reliable cost and schedule estimates for software development projects is still largely a dream. Software development cost is notoriously difficult to estimate reliably at an early stage.

Software Engineering: Principles and Practice.

Hans van Vliet, Wiley, 2007.

The Cloud

Cost savings as motivation for cloud adoption

- Converting CAPEX to OPEX
  - No upfront hardware procurement costs
- Race to Zero in some types of offerings e.g. storage
  - Vendor lock-in as a side-effect

Public cloud provider cost models

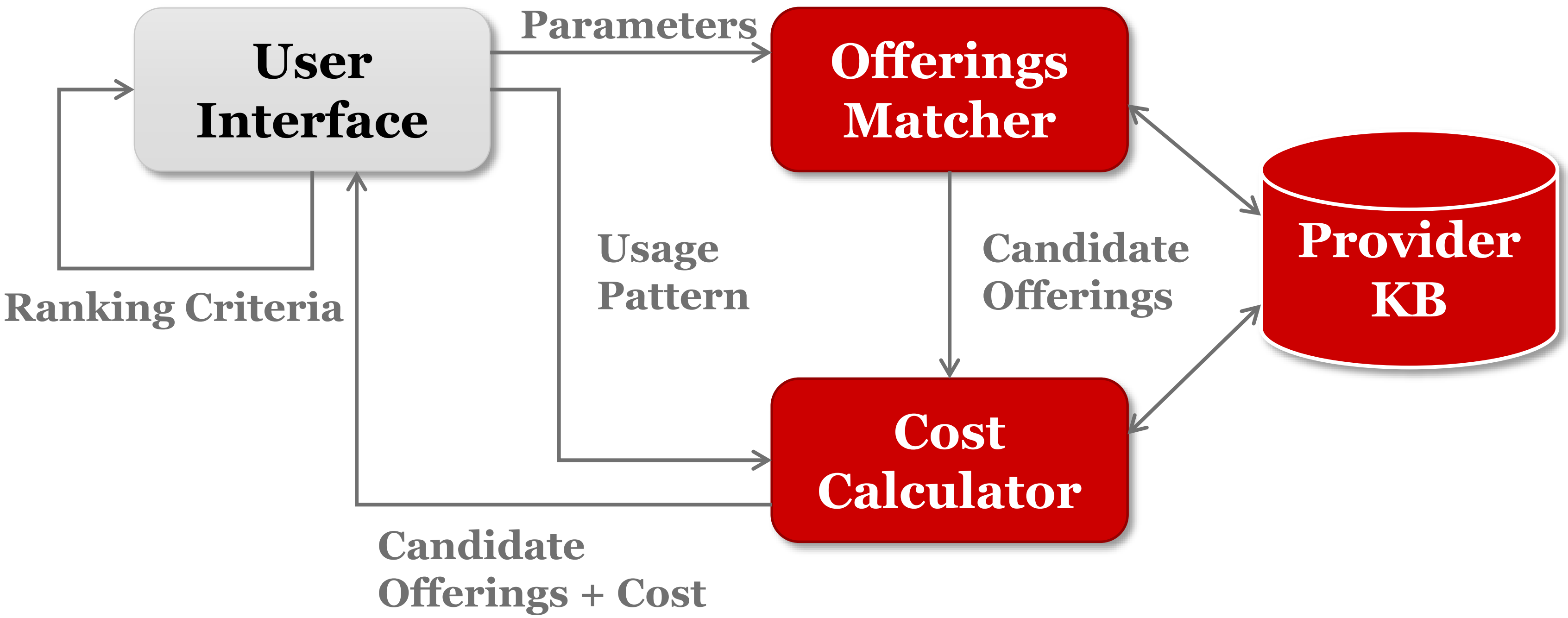
- Cost as tiered linear functions of time/volume [Andrikopoulos et al. 2013]

$$f(x,y) = \begin{cases} (0.095 \times \frac{x}{1000}) + \frac{(0.01 \times y)}{100000} & x \in [0,1] \\ (95 + (0.8 \times \frac{x-1}{1000}) + \frac{(0.01 \times y)}{100000} & x \in (1,50] \\ (4015 + (0.7 \times \frac{x-50}{1000}) + \frac{(0.01 \times y)}{100000} & x \in (50,500] \\ \ldots \end{cases}$$

- Billable Time Units (BTUs) have gone down from hour(s) to seconds in the last years

- Prices per hour/GB are continuing to fall rapidly

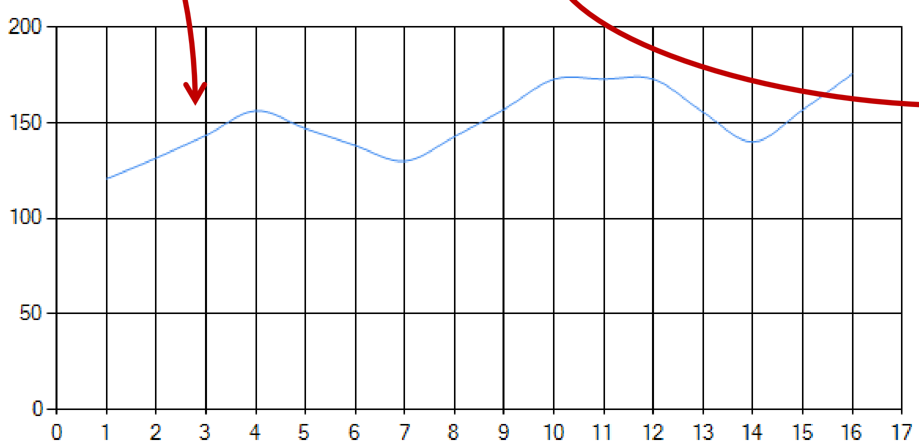
Cloud service selection decision support



Services ranking

Configuration | Selection Results | Ranking Results

	Configuration	Location	Detail	Provider	Cost	Sum	Parameter	
☐	n1-standard-8-d	North America	Per Month	Google	110.4 $	2207.22 $	Storage: 3540 Memory: 30 cpuCores: 8	Info
☐	n1-standard-8-d	Europe	Per Month	Google	120.8 $	2414.96 $	Storage: 3540 Memory: 30 cpuCores: 8	Info
☐	n1-highmem8-d	North America	Per Month	Google	127.2 $	2543.08 $	Storage: 3540 Memory: 52 cpuCores: 8	Info
☐	n1-highmem8-d	Europe	Per Month	Google	143.2 $	2862.79 $	Storage: 3540 Memory: 52 cpuCores: 8	Info
☐	ExtraLarge	worldwide	Per Month	Microsoft	64 $	1279.62 $	Storage: Memory: 14 cpuCores: 8	Info



Offering	ComputeEngine
Configuration	n1-standard-8-d
cpuCores	8
cpuSpeed	2.6
IO	
Memory	30
Platform	64
SLA	0.9995
Storage	3540

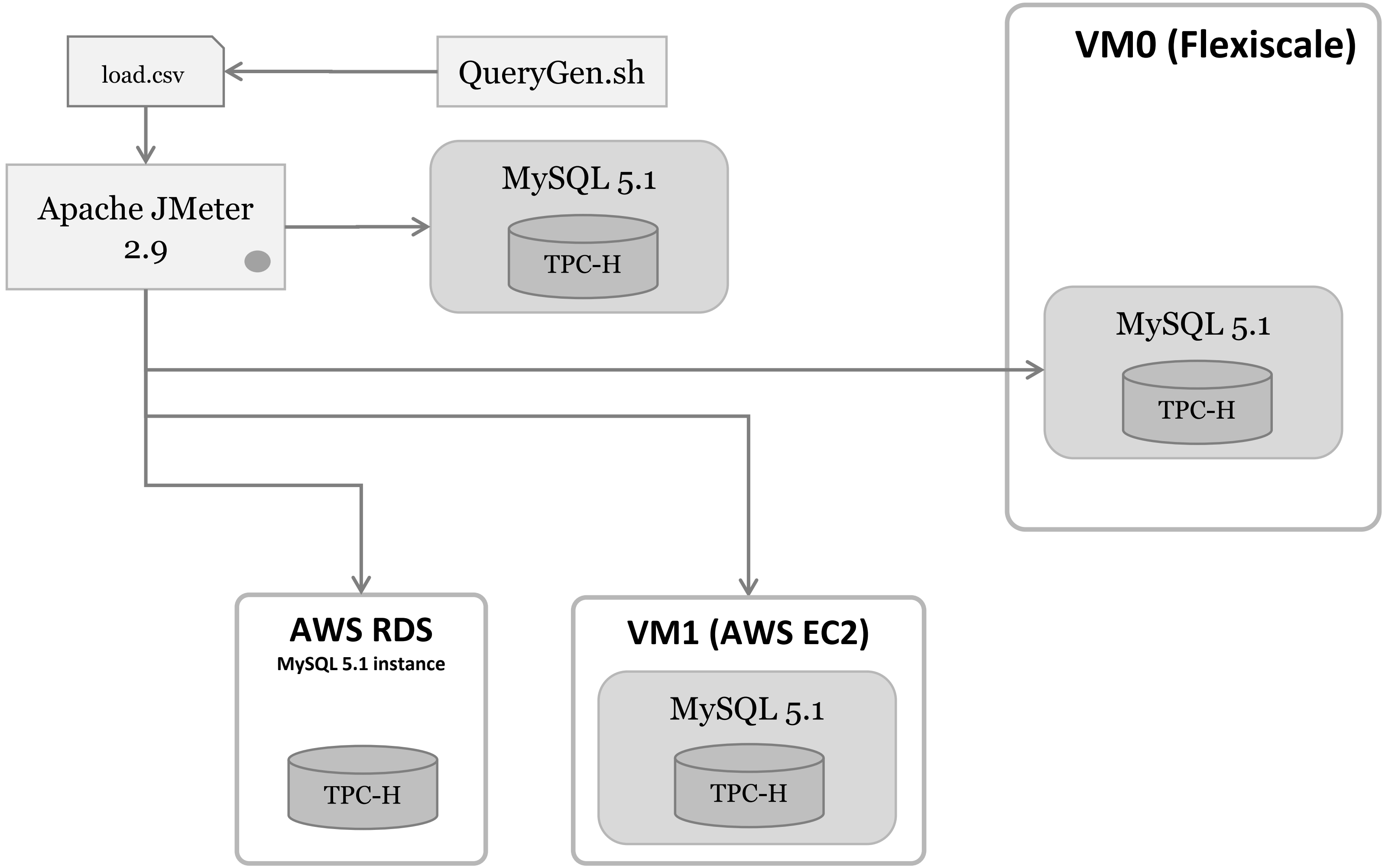
Ranking As

Cost

Select All | Reset | Next Step

university of groningen

Application performance x deployment



[Gómez Sáez et al. 2014]

Load characterization

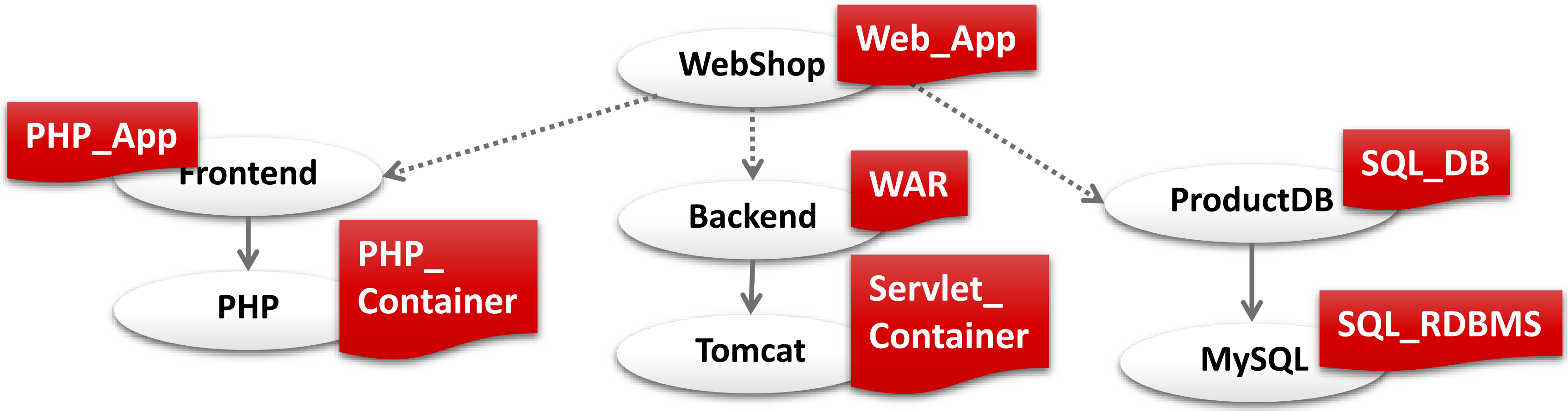
Table I: TPC-H Workload Analysis.

Query	Accessed Tables	Subqueries	Total Logical Evaluations	Throughput (Req./s) On-Premise	Throughput (Req./s) IaaS Flexiscale	Throughput (Req./s) IaaS AWS EC2	Throughput (Req./s) DBaaS	Retrieved Data (B)	Category ID
$Q^{(1)}$	1	0	1	0.03425	0.03396	0.04115	0.03817	538	CH
$Q^{(2)}$	5	1	13	0.07927	0.14884	0.07413	3.03260	15857	CH
$Q^{(4)}$	2	1	5	0.53950	0.73922	0.54244	0.94903	105	CL
$Q^{(5)}$	6	0	9	0.01148	0.02014	0.01377	0.33484	130	CH
$Q^{(6)}$	1	0	4	0.20583	0.21355	0.22450	0.28261	23	CL
$Q^{(7)}$	5	1	11	0.03123	0.04782	0.03477	0.20792	163	CH
$Q^{(8)}$	7	1	11	0.97156	1.45380	0.74072	0.18196	49	CM
$Q^{(9)}$	6	1	8	0.05947	0.09123	0.05470	0.05548	4764	CH
$Q^{(10)}$	4	0	6	0.09168	0.11970	0.09584	0.49834	3454	CH
$Q^{(11)}$	3	1	6	2.59998	4.07134	1.85092	0.26802	16069	CL
$Q^{(12)}$	2	0	7	0.21147	0.22465	0.23487	0.13981	71	CL
$Q^{(13)}$	2	1	2	0.12771	5.32350	-	-	16	CL
$Q^{(14)}$	2	0	3	0.03373	0.06017	0.03444	0.29052	28	CH
$Q^{(15)}$	1	0	2	201.53365	22.25911	12.0840	23.11528	9	CL
$Q^{(16)}$	2	1	2	0.11346	0.11219	0.12755	0.13471	120	CM
$Q^{(17)}$	3	1	6	0.10931	0.19021	0.11319	0.97148	648259	CL
$Q^{(18)}$	2	1	5	0.98213	1.81212	-	-	25	CL
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
$Q^{(20)}$	2	0	25	4.05648	4.90228	3.29667	0.17083	21	CM
$Q^{(21)}$	[illegible]	2	8	[illegible]	[illegible]	[illegible]	-	[illegible]	CM
$Q^{(22)}$	4	2	13	0.01070	0.01734	0.01610	0.06065	8944	CH
$Q^{(23)}$	2	2	6	2.72083	3.30785	2.35940	-	137	CL

1K queries with synthetic load

Scenario	Category	% Queries same Category	Distribution Parameters	Throughput (Req./s)
On-Premise	CL	79.4%	k= 0.35666 λ= 3.28983	0.27749
On-Premise	CM	18.9%	k= 0.36037 λ= 0.80655	0.05888
On-Premise	CH	95.0%	k= 0.53023 λ= 0.08990	0.02696
DBaaS	CL	66%	k= 0.54324 λ= 0.59264	0.45238
DBaaS	CM	21.6%	k= 0.57200 λ= 0.88472	0.19972
DBaaS	CH	88.3%	k= 0.74471 λ= 0.23991	0.10273
IaaS	CL	78.2%	k= 0.63816 λ= 1.64010	0.34477
IaaS	CM	20.0%	k= 0.52690 λ= 0.53472	0.06046
IaaS	CH	90.8%	k= 0.60906 λ= 0.11362	0.03378

Application Topology



An **application topology** is a labeled graph

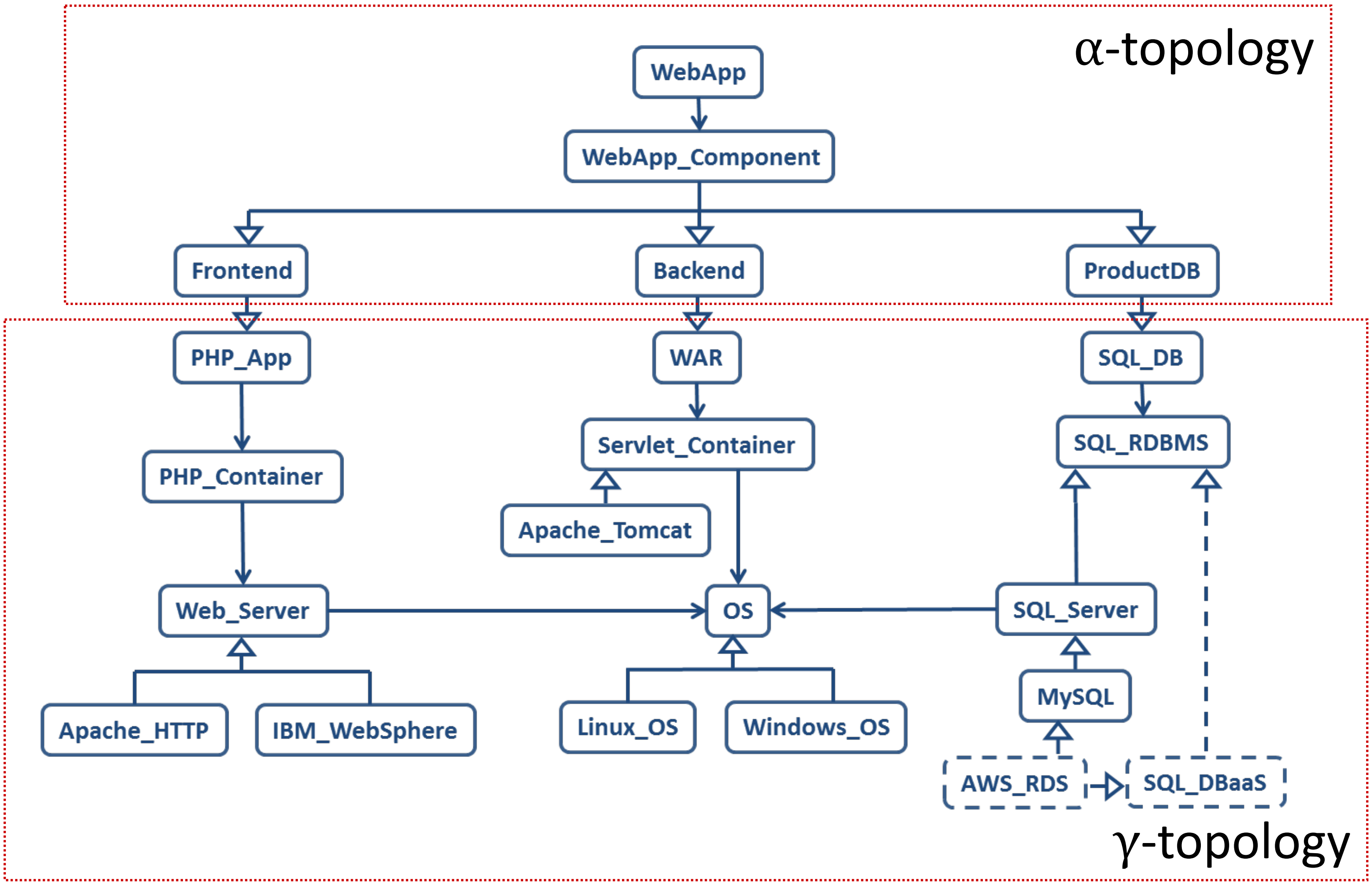
$$G = (N^L, E^L, s, t)$$

where $N: nodes, E: edges, L: labels$ & $s, t: E^L \rightarrow N^L$ [Andrikopoulos et al 2014]

The topology graph T is called **typed** if the label set contains **only** typed elements.

A (typed) topology T is **viable** w.r.t. a type graph with inheritance TG_I iff **all** elements of T are *labeled* over the elements of TG_I. TG_I is then called the **μ-topology** of the application.

Example of a μ-topology

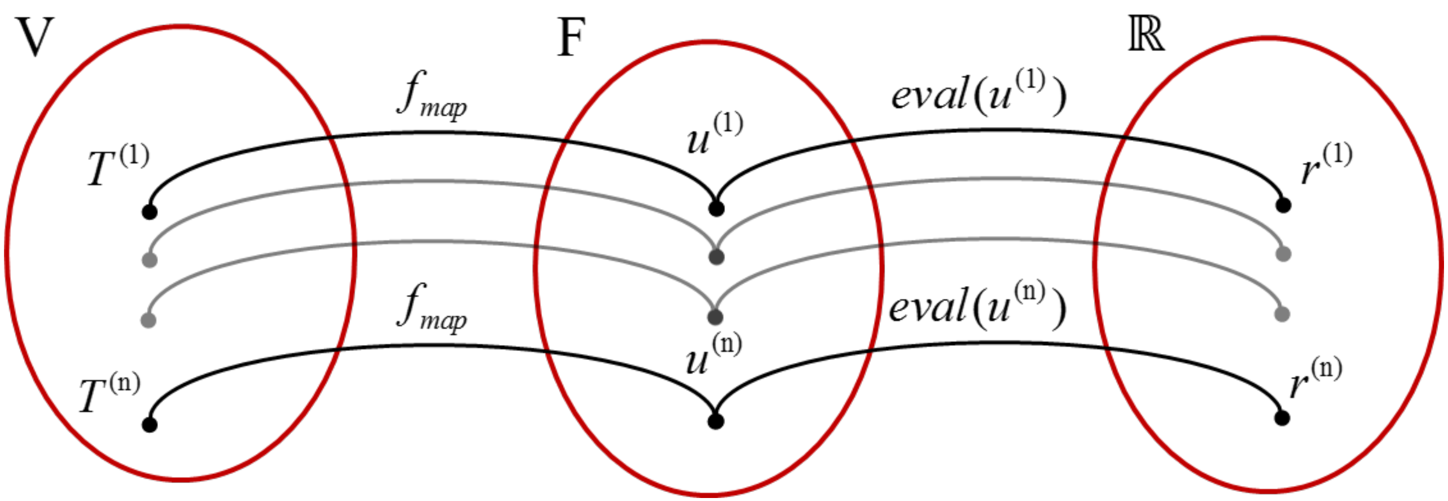


Utility Function

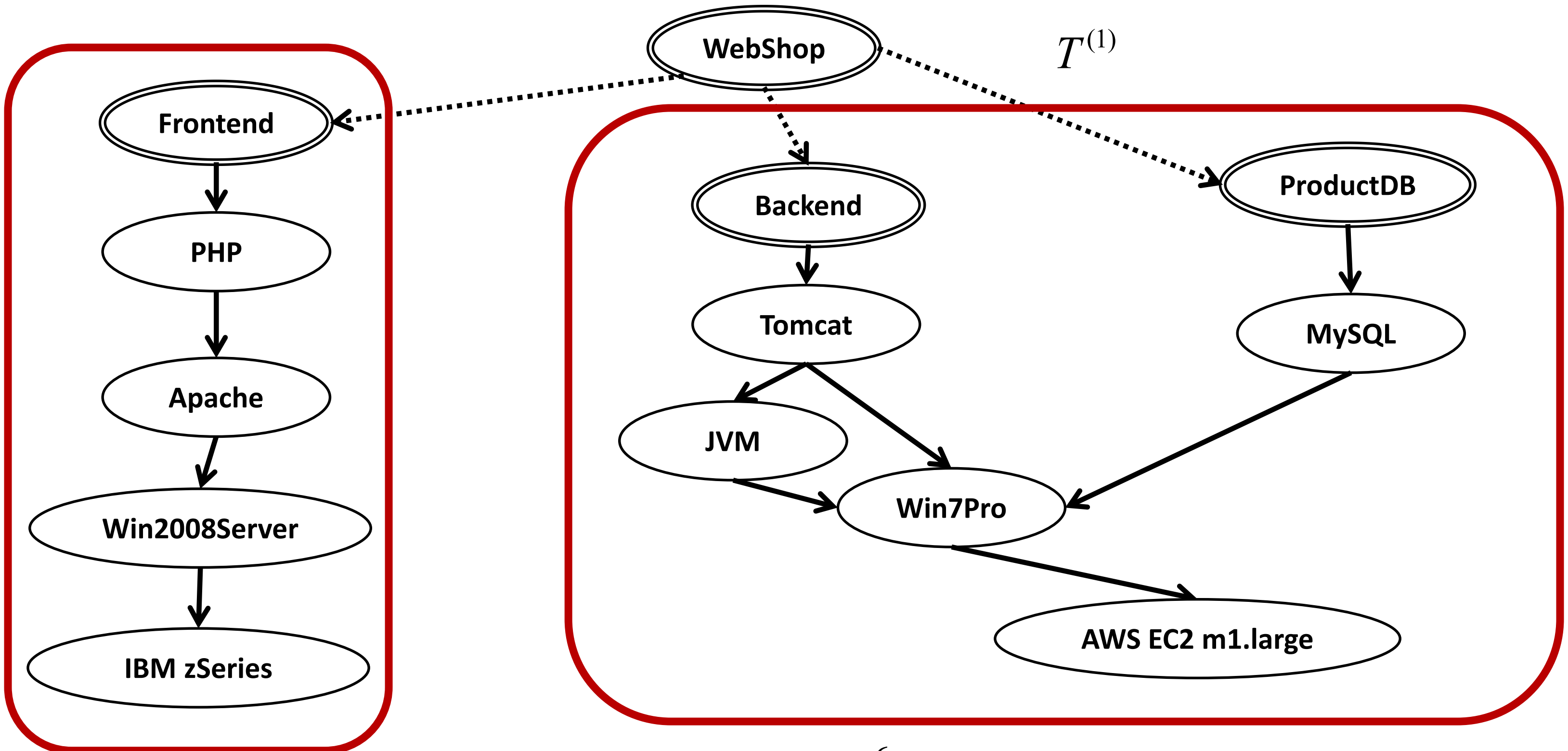
- Set of functions $\boldsymbol{F}$ on real numbers $\mathbb{R}$ such as:

$$\boldsymbol{F} = \{f(a_1, \dots, a_n) | n \geq 1, f: \mathbb{R}^* \to \mathbb{R}\}$$

- Mapping function $f_{map}: \boldsymbol{V} \to \boldsymbol{F}$ from $\boldsymbol{F}$ to set of all viable topologies $\boldsymbol{V}$
- Utility function: $u^{(i)}(a_1, \dots, a_n) = f_{map}\left(T^{(i)}\right) = u^{(i)}(T^{(i)})$
- Given a set of values $(p_1, \dots, p_n)$ then the function can be evaluated as $r^{(i)} = eval\left(u^{(i)}(T^{(i)}), (p_1, \dots, p_n)\right) = u^{(i)}(p_1, \dots, p_n), r^{(i)} \in \mathbb{R}$



Utility based on Operational Expenses (OPEX)



$$opex_{zSeries}(h_\tau, z) = k_e \times h_\tau \times z$$

[Walker 2009]

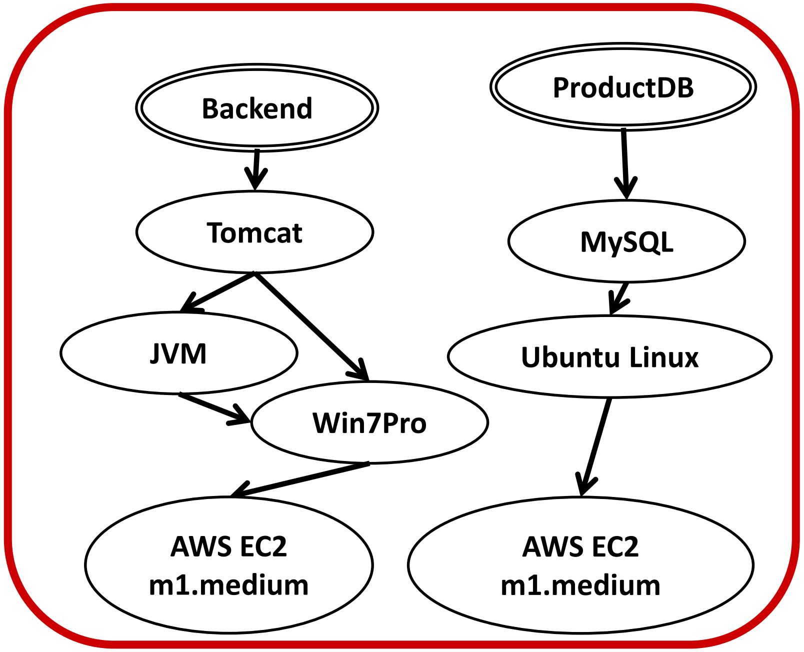
$$opex_{EC2\_m1.large}(h_\tau) = \begin{cases} 243 + h_\tau \times 0.17 & \text{with 1-year contract} \\ 384 + h_\tau \times 0.134 & \text{with 3-year contract} \end{cases}$$

[Andrikopoulos et al 2013]

$$u(T^{(1)}) = u^{(1)}(k_{max}, h_\tau, z) = k_{max} - (opex_{zSeries}(h_\tau, z) + opex_{EC2\_m1.large}(h_\tau))$$

OPEX-oriented optimization

$T^{(2)}$



...

$T^{(n)}$

$u(T^{(n)}) = \ldots$

$$u(T^{(2)}) = k_{max} - (opex_{zSeries}(h_\tau, z) + opex_{EC2\_m1.medium(Win)}(h_\tau) + opex_{EC2\_m1.medium(Lnx)}(h_\tau))$$

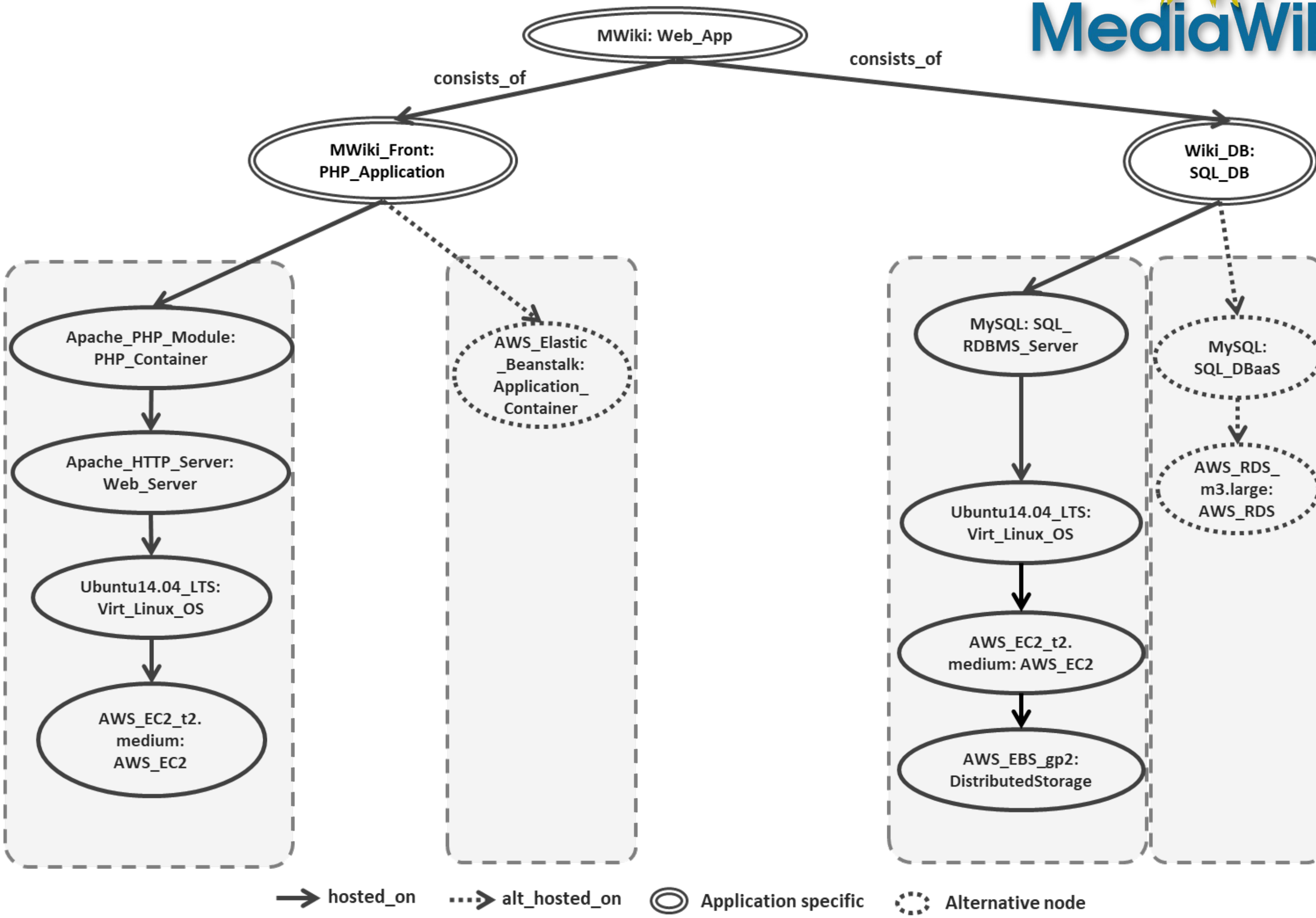
$$u^{(i)}(h_\tau, \tau, n_{I/O}, d_{storage}, d_{egress}, loc) = opex_{max} - opex^{(i)}(h_\tau, \tau, \ldots)$$

[Andrikopoulos et al 2014]

university of groningen

The MediaWiki Case Study

MediaWiki

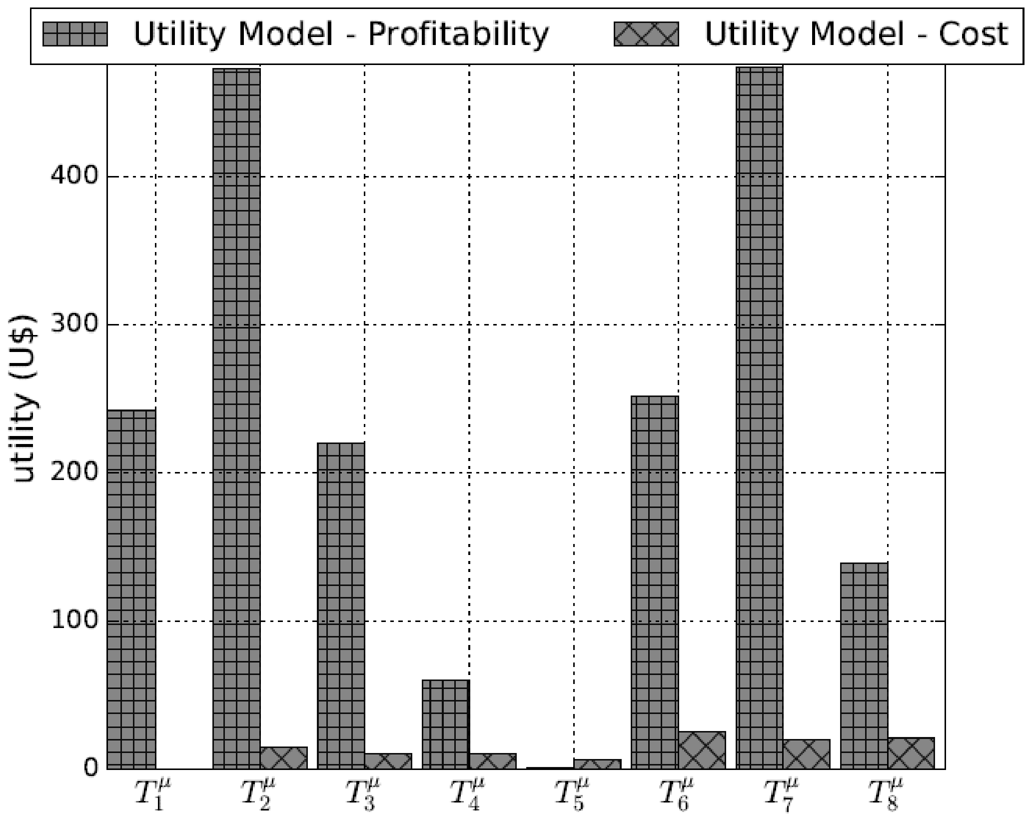


Cost vs Profitability

Table I: Evaluation Setup - Viable Distributions (T^{μ}) of the MediaWiki Application. *Prices are calculated for the on-demand usage. Storage and data transfer costs are billed separately.*

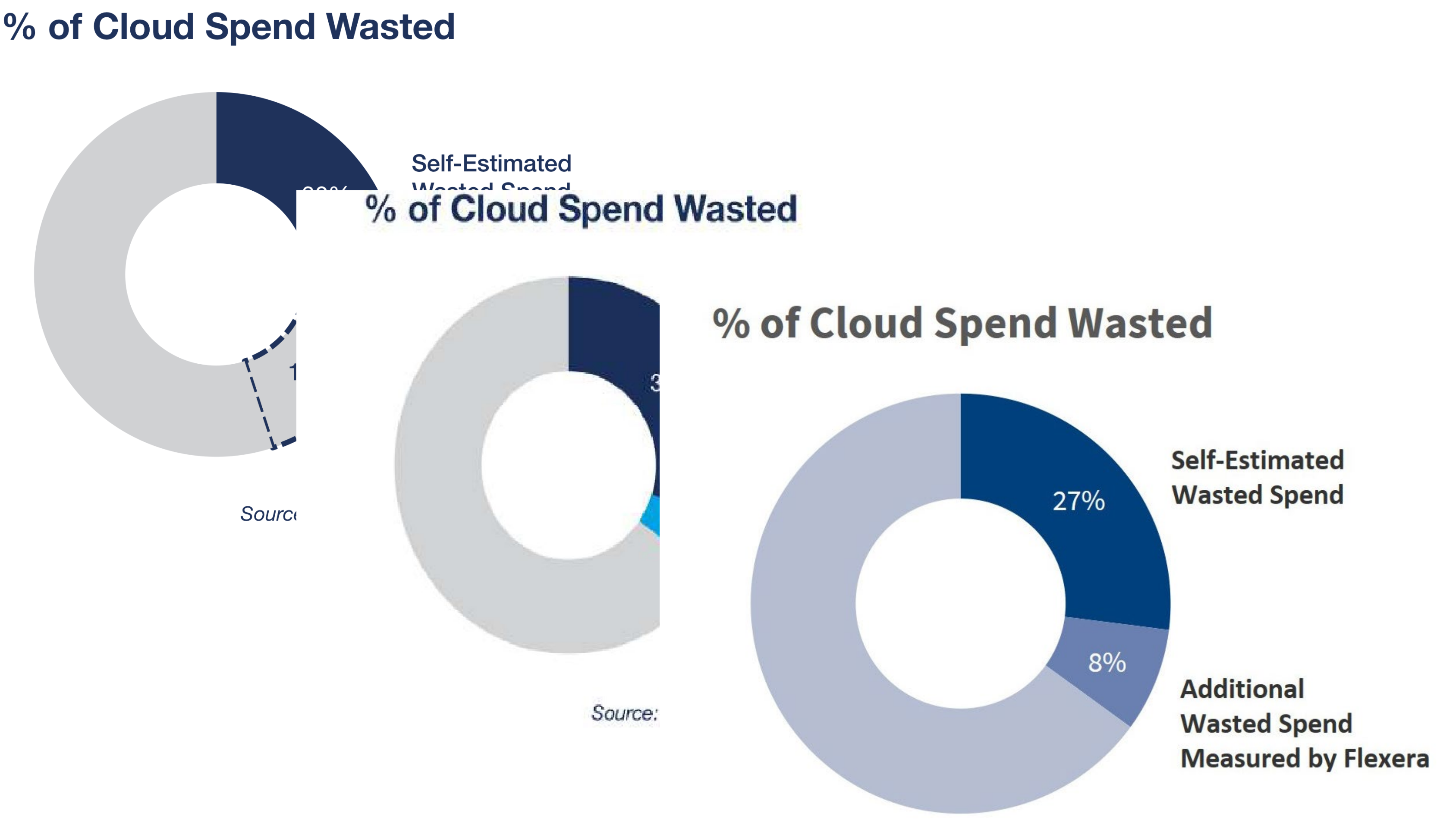
T^{μ}	Service	MediaWiki Front-end	MediaWiki Back-	Auto-	Region	Total Price
T^{μ}_1	EC2	m4.large	m4.la			
T^{μ}_2	EC2	m4.xlarge				
T^{μ}_3	EC2 + RDS	m4.large	db.m4.			
T^{μ}_4	Beanstalk	(2x) t2.small	db.m4.			
T^{μ}_5	ECS	(2x) t2.small	db.m4.			
T^{μ}_6	VM	DS2	DS			
T^{μ}_7	VM	DS3				
T^{μ}_8	Container	(2x) DS1	DS			

☐ Amazon Web Services ☐ Windows



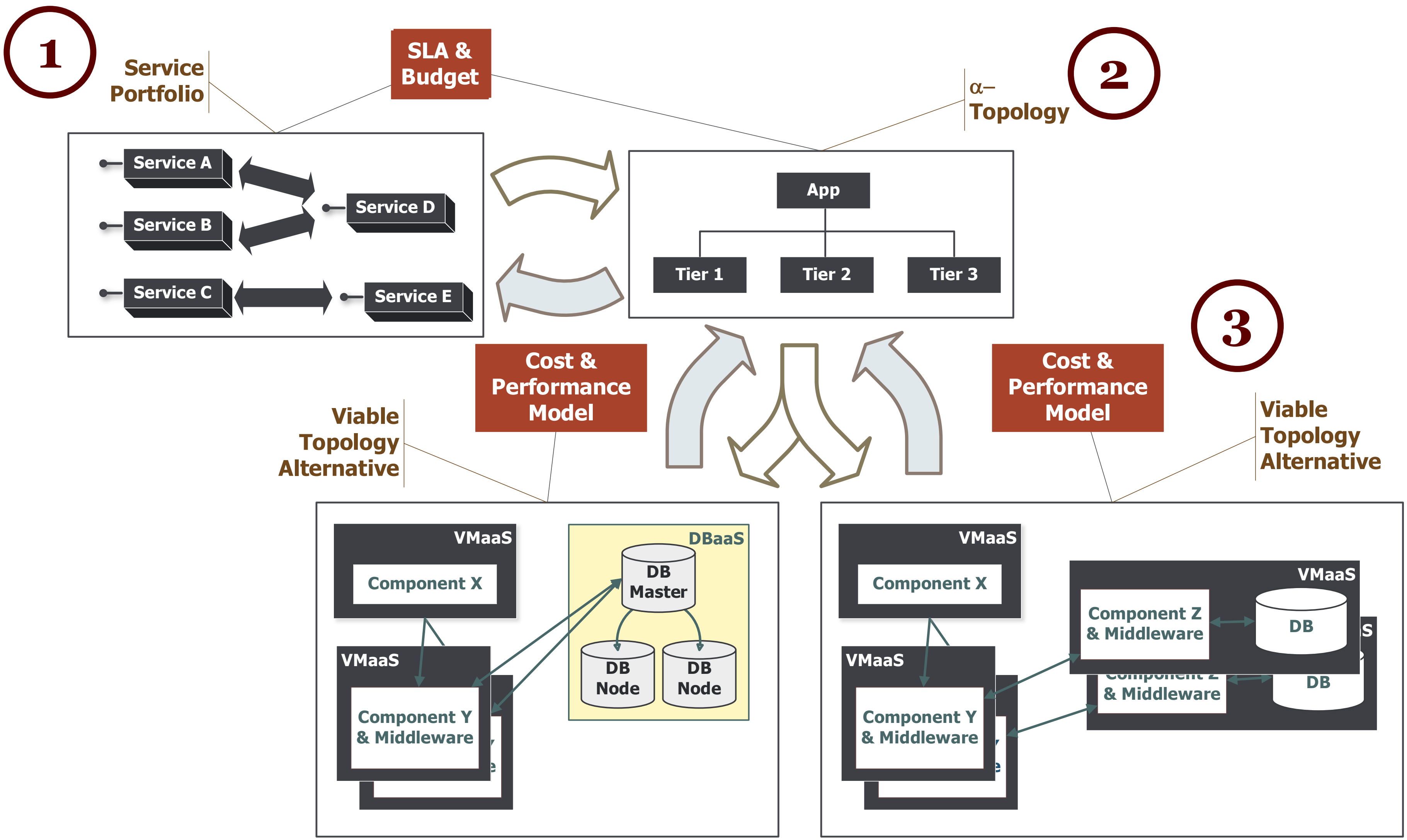
[Gómez Sáez et al. 2018]

Computational waste in the cloud

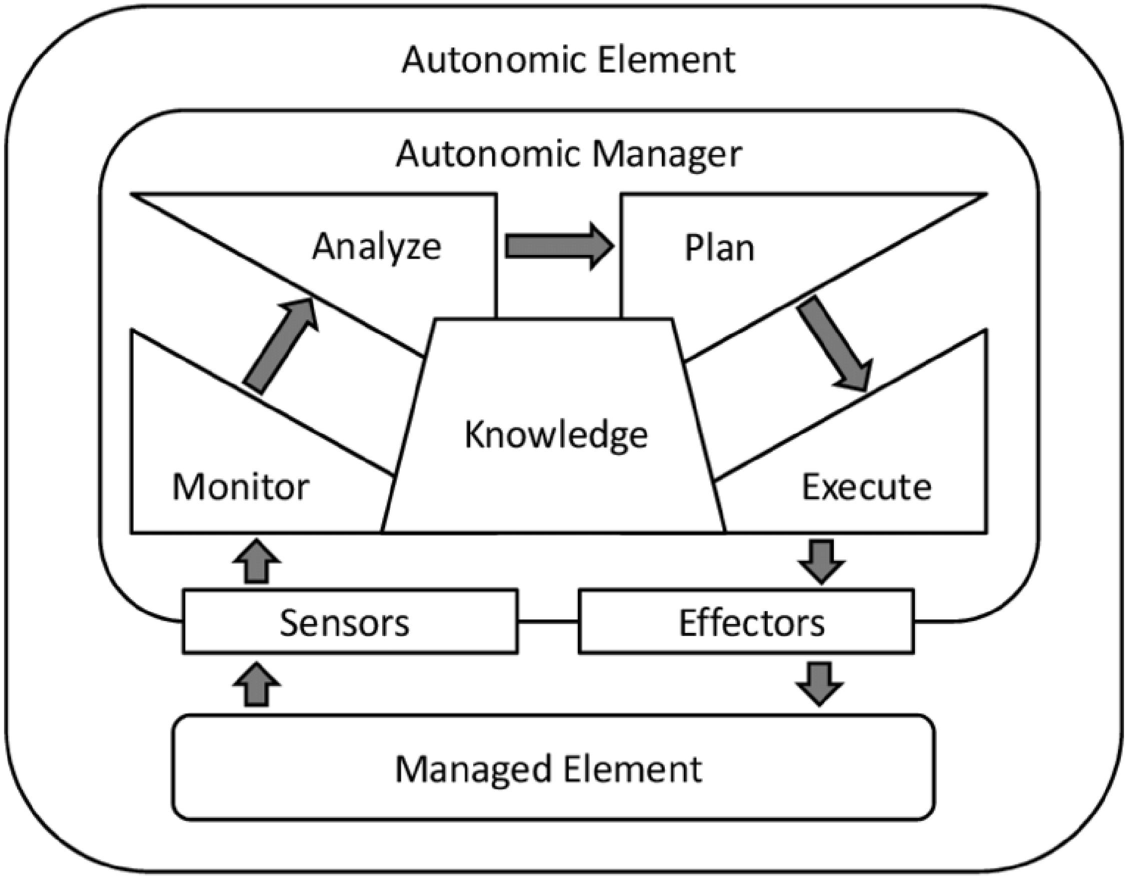


Source: RightScale 2019 State of the Cloud Report from Flexera

The Cloud-Based Application (CBA) lifecycle

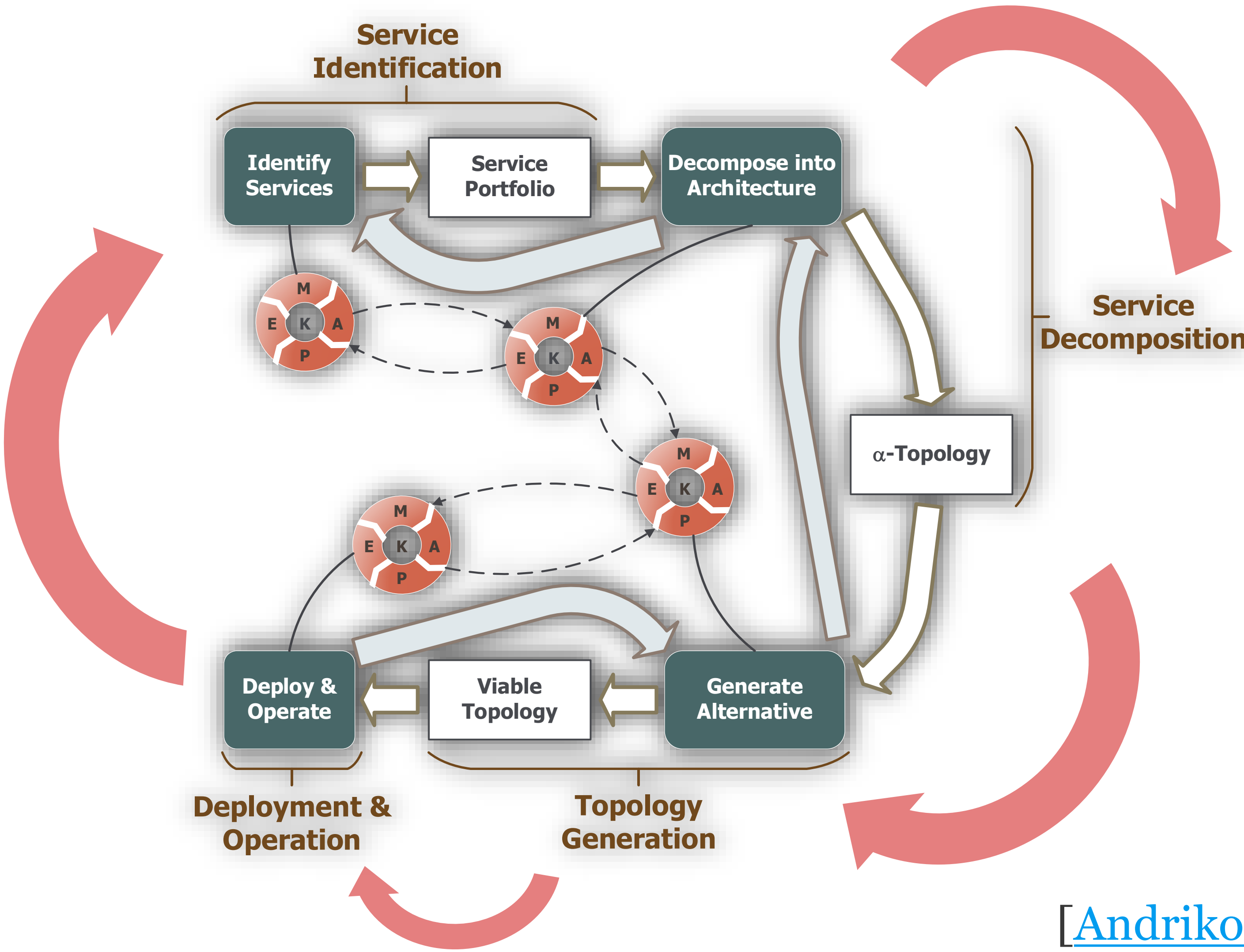


MAPE-K



[IBM 2005]

Phases of the CBA lifecycle



[Andrikopoulos 2017]

Cost efficiency in software (re)architecting

- Multiple goals
  - Decrease cost without hurting performance (underprovisioning)
  - Minimize waste (overprovisioning)
- During design time
  - Informed decision making during design cycles
  - Cost/performance trade-offs under consideration
- During run time
  - Active monitoring, refactoring, and redeployment of the application

Main take-aways

- Cost management as an architecting activity
  - Counter-intuitive phenomena
  - Experimentation/monitoring is required
- Design- and run-time activities are complementary, not mutually exclusive

Reach out to me at:

v.andrikopoulos@rug.nl
https://vandriko.github.io
@v_andrikopoulos

university of groningen

Further reading

- Andrikopoulos, V., Binz, T., Leymann, F., & Strauch, S. (2013). How to adapt applications for the cloud environment. *Computing*, *95*(6), 493-535.

- Andrikopoulos, V., Reuter, A., Xiu, M., & Leymann, F. (2014, June). Design support for cost-efficient application distribution in the cloud. In *2014 IEEE 7th International Conference on Cloud Computing* (pp. 697-704). IEEE.

- Sáez, S. G., Andrikopoulos, V., Hahn, M., Karastoyanova, D., Leymann, F., Skouradaki, M., & Vukojevic-Haupt, K. (2015, May). Performance and cost trade-off in IaaS environments: a scientific workflow simulation environment case study. In *International Conference on Cloud Computing and Services Science* (pp. 153-170). Springer, Cham.