Cloud-Native Databases: An Application Perspective

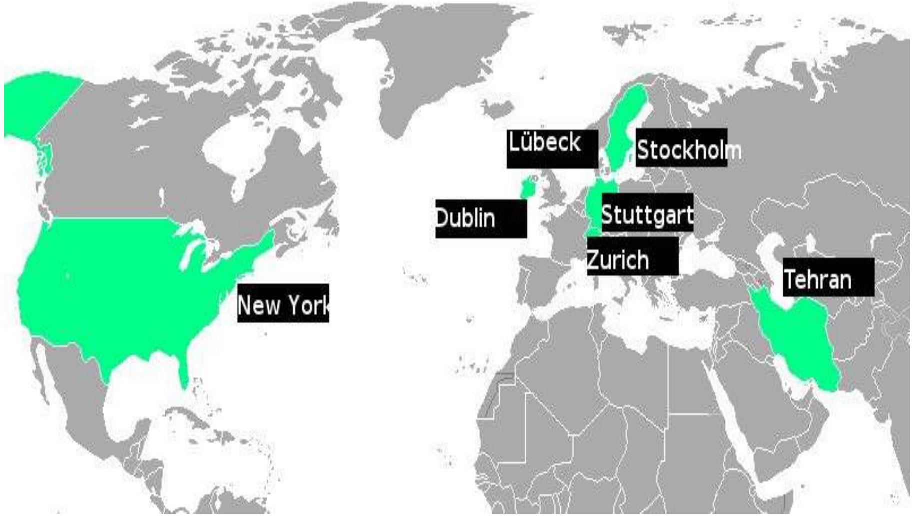
Josef Spillner
Giovanni Toffetti
Manuel Ramírez López (ramz@zhaw.ch)

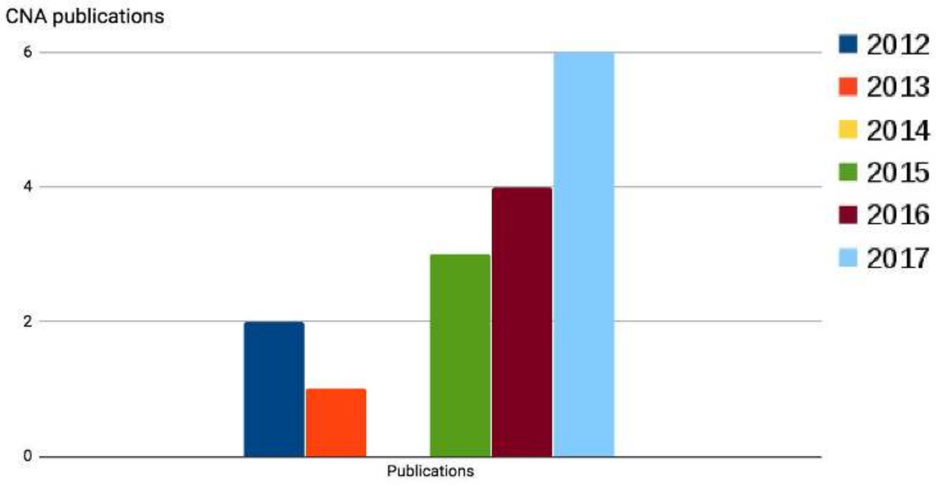
3rd CloudWays Workshop
27/09/2017

Service Engineering Research Area
ICCLAB
SPLAB

Zurich University of Applied Sciences
zh
aw

Introduction





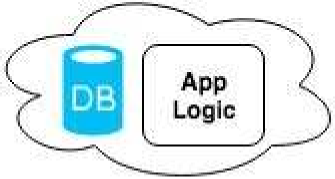
Cloud Native Applications/Architecture

Motivation

“Cloud-Native Database”

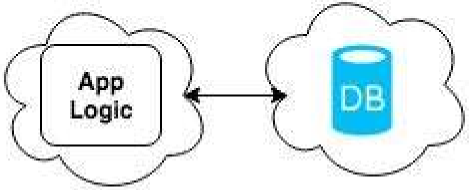
Self-managed database service

- More control of the database
- More multi-tenancy options
- More effort in creation, configuration and deployment
- Closer to the logic of the app



Provider-managed database services (DBaaS)

- Less control of the database
- Not all the multi-tenancy options
- Less effort in creation, configuration and deployment
- Attractive cloud pricing



Our studies

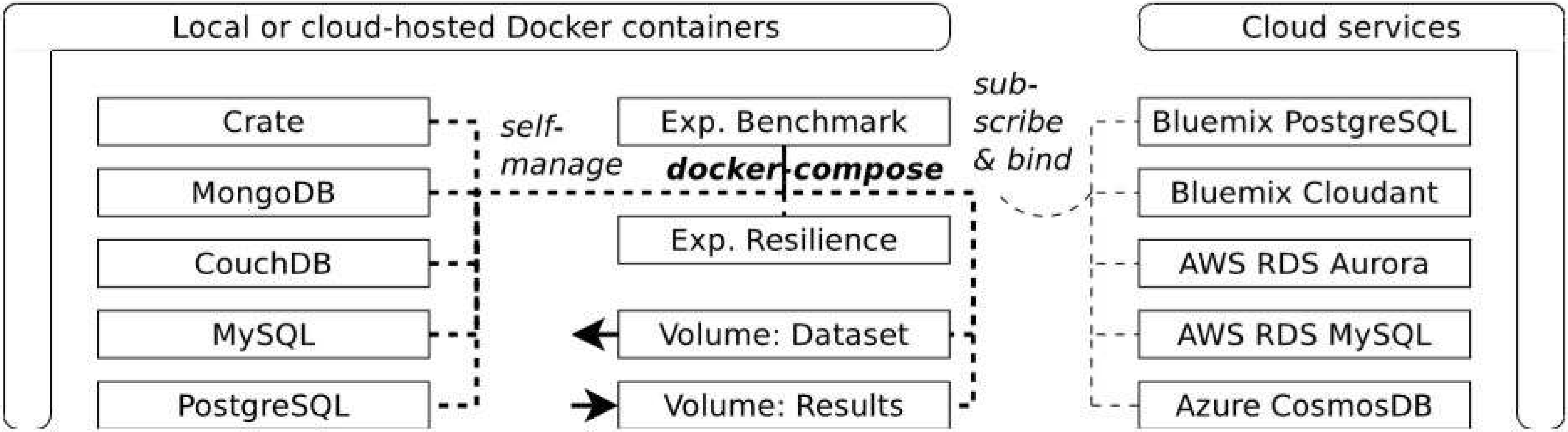
Properties:

- Performance
- Pricing
- Multitenancy
- Resilience
- Scalability

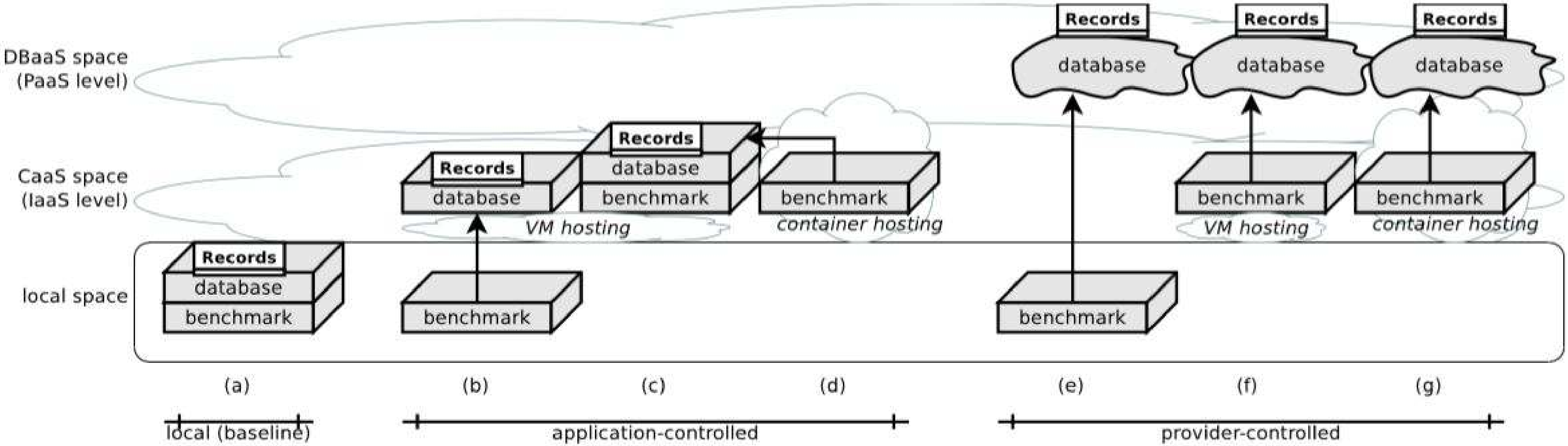
Method:

- Experimental method
- Measurement
- Systematic
- Repeatable

Testbed Architecture

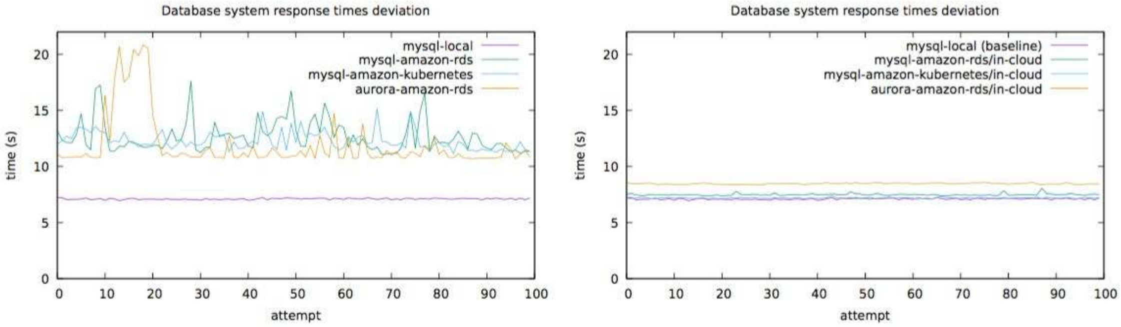


Experiments

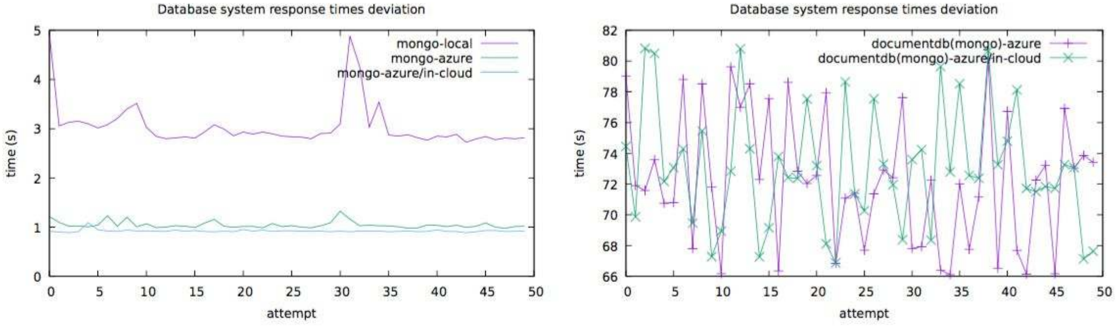


Paper	Appendix	Open science notebook
7	5	28 (raw results)

Performance results

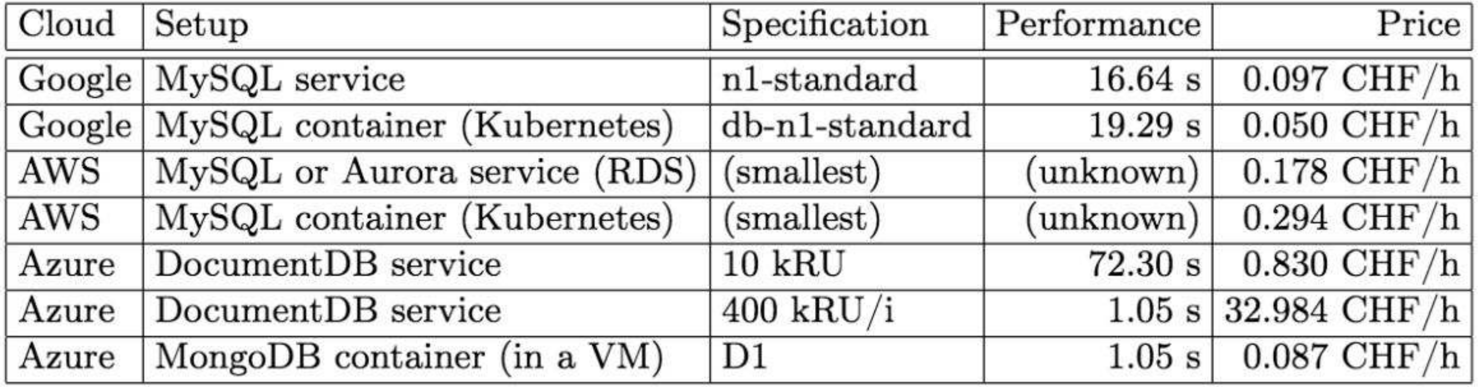


(a) Local benchmark (b) Cloud benchmark



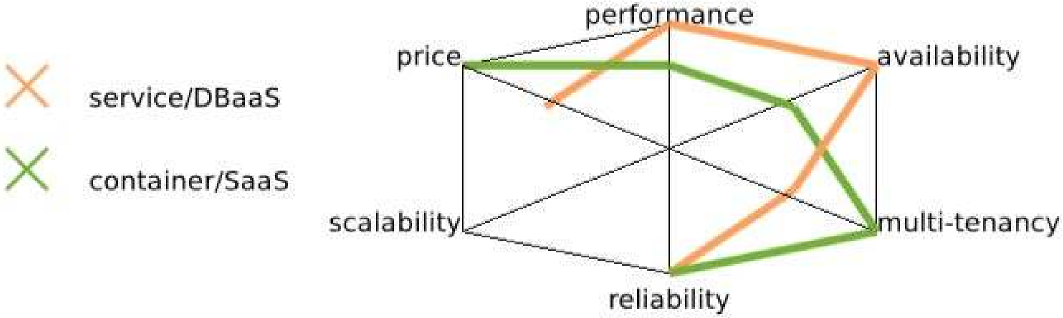
(a) MongoDB self-managed instances (b) DocumentDB with MongoDB interface

Pricing results (not quantified)

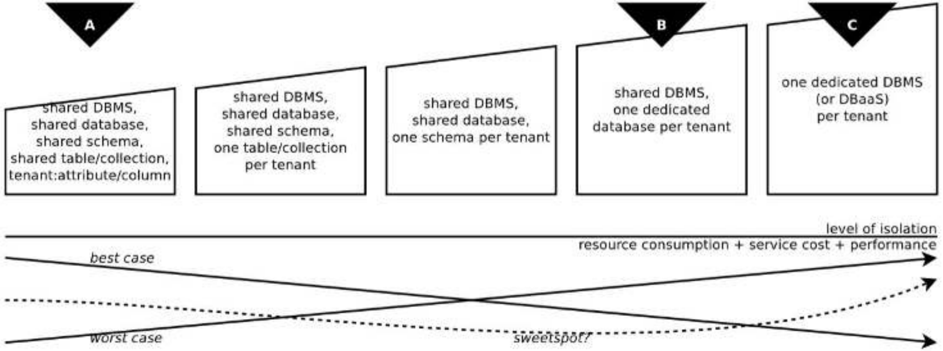


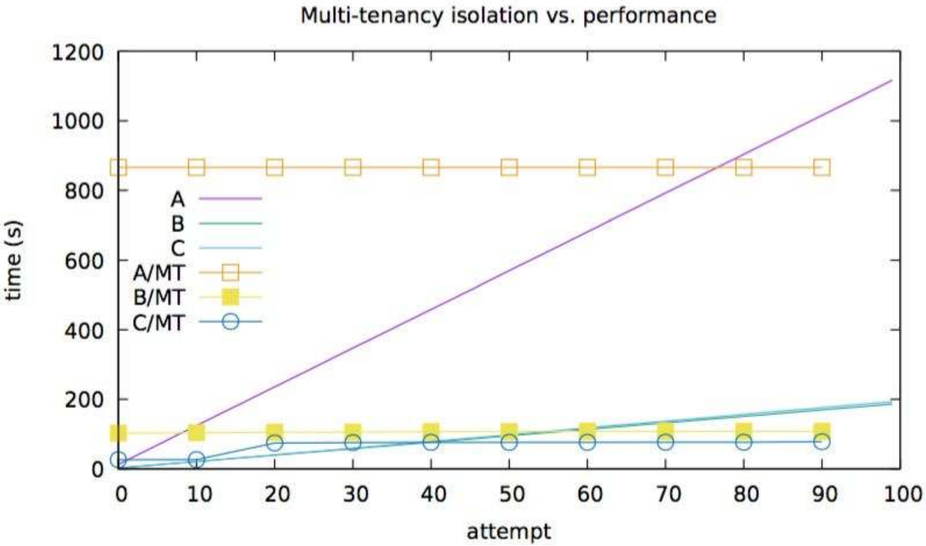
Cloud	Setup	Specification	Performance	Price
Google	MySQL service	n1-standard	16.64 s	0.097 CHF/h
Google	MySQL container (Kubernetes)	db-n1-standard	19.29 s	0.050 CHF/h
AWS	MySQL or Aurora service (RDS)	(smallest)	(unknown)	0.178 CHF/h
AWS	MySQL container (Kubernetes)	(smallest)	(unknown)	0.294 CHF/h
Azure	DocumentDB service	10 kRU	72.30 s	0.830 CHF/h
Azure	DocumentDB service	400 kRU/i	1.05 s	32.984 CHF/h
Azure	MongoDB container (in a VM)	D1	1.05 s	0.087 CHF/h

Spider graph for pricing trade-offs.
Sampled for MySQL at Google.
Outside = best.



Multitenancy results


A
B
C
shared DBMS, shared database, shared schema, shared table/collection, tenant:attribute/column
shared DBMS, shared database, shared schema, one table/collection per tenant
shared DBMS, shared database, one schema per tenant
shared DBMS, one dedicated database per tenant
one dedicated DBMS (or DBaaS) per tenant
level of isolation
resource consumption + service cost + performance
best case
worst case
sweetspot?



Findings and recommendations

FINDINGS	RECOMMENDATIONS
Determining the best database is not possible	CNDBBench tool
Limitations: ● Crate (return 10000 rows by default) ● CosmosDB (1000 RU per seconds) ● PyMongo: 20 seconds query timeout in inserting many records	● Discoverable description of these properties ● More complete documentation

For future applications (in more mature containerised database systems)

↓

auto-clustering microservices (as Crate)

Conclusions

- Identified the **different options** and the key **properties** of a Cloud Native Database.
- Created a **method** (with a **tool**) to help to compare all the properties in the different options.
- Open question: What is better?

Repeatability

https://github.com/serviceprototypinglab/cndbbench

Benchmark

https://github.com/serviceprototypinglab/cndbresults

Open science notebook & results