Cloud-Native Databases: An Application Perspective

Josef Spillner
Giovanni Toffetti
Manuel Ramírez López (ramz@zhaw.ch)
3rd CloudWays Workshop
27/09/2017
Introduction

Cloud Native Applications/Architecture
## Motivation

### “Cloud-Native Database”

<table>
<thead>
<tr>
<th>Self-managed database service</th>
<th>Provider-managed database services (DBaaS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>● More control of the database</td>
<td>● Less control of the database</td>
</tr>
<tr>
<td>● More multi-tenancy options</td>
<td>● Not all the multi-tenancy options</td>
</tr>
<tr>
<td>● More effort in creation, configuration and deployment</td>
<td>● Less effort in creation, configuration and deployment</td>
</tr>
<tr>
<td>● Closer to the logic of the app</td>
<td>● Attractive cloud pricing</td>
</tr>
</tbody>
</table>

[Diagram showing the difference between self-managed and provider-managed database services]
Our studies

Properties:
- Performance
- Pricing
- Multitenancy
- Resilience
- Scalability

Method:
- Experimental method
- Measurement
- Systematic
- Repeatable
Testbed Architecture
# Experiments

<table>
<thead>
<tr>
<th>Paper</th>
<th>Appendix</th>
<th>Open science notebook</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>5</td>
<td>28 (raw results)</td>
</tr>
</tbody>
</table>
Performance results

(a) Local benchmark
(b) Cloud benchmark

(a) MongoDB self-managed instances
(b) DocumentDB with MongoDB interface
## Pricing results (not quantified)

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

Spider graph for pricing trade-offs. Sampled for MySQL at Google. Outside = best.
Multitenancy results
# Findings and recommendations

<table>
<thead>
<tr>
<th>FINDINGS</th>
<th>RECOMMENDATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determining the best database is not possible</td>
<td>CNDBBench tool</td>
</tr>
</tbody>
</table>

**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

Open science notebook & results

https://github.com/serviceprototypinglab/cndbresults