Predictable elasticity of Docker applications

Manuel Ramírez López
Service Prototyping Lab. ZHAW
ramz@zhaw.ch
31/05/2017
14th Docker Switzerland User Group Meetup
Application composed of multiple Docker containers

Example of docker app

Microservices architecture
Motivation

Why?

- The replicas are stealing resources from other microservices which are on the critical performance path.
- The ineffective scaling containers which are not the bottleneck of the application.
How do Docker containers scale?

**Stateful docker container**: A container which maintains state locally across invocations. So must to handle the persistence of the state. Examples: databases and message queues.

**Stateless docker container**: Do not keep any state and therefore do not persist data except through other services.
How do Docker containers scale?

Scale-cube

- **X-axis**: horizontal duplication. Scale by cloning.
- **Z-axis**: data partitioning. Scale by splitting similar data structures.
- **Y-axis**: functional decomposition. Scale by splitting different functionality.
Autoscaling

1. Some microservices do not scale out or in as fast as is needed. A prediction of when they need to scale is necessary to achieve elasticity.

2. Not all microservice implementations can be auto-scaled by instantiation alone. Stateful services are often not recognised automatically.

3. Auto-scaling over the top of a bottleneck is in vain.
Some concepts

What is the best combination?

- Combination = \((n_1, n_2, \ldots, n_k)\)

The 3 factors:
- Use case (fix)
- Cost
- Performance

Example combination: 3 CRUD replicas, 1 database replica (3,1)
Scalability formula

● What is the most economical combination satisfying minimum performance constraints?

● What is the fastest combination satisfying maximum price constraints?
Step 1: Create a performance matrix

And now some math to show how that works.
Step 2: Obtain a solution

\[ \text{fastest}(M_e, \text{prices}, \max_{\mu}, \max_{\kappa}) = i \mid \min_{\forall i \in I} \{m_i \in M_e \mid m_i < \max_{\mu}, \text{cost}(i, \text{prices}) < \max_{\kappa}\} \quad (1) \]

\[ \text{cheapest}(M_e, \text{prices}, \max_{\mu}, \max_{\kappa}) = i \mid \min_{\forall i \in I} \{\text{cost}(i, \text{prices}) \mid M_e \ni m_i < \max_{\mu}, \text{cost}(i, \text{prices}) < \max_{\kappa}\} \quad (2) \]

\[ \text{fastest\_rate}(M_e, \text{prices}, \max_{\mu}, \max_{\kappa}, \text{rate}) = i \mid \min_{\forall i \in I} (i \mid \frac{m_i}{m_k} \leq \text{rate}, \prec_{\text{cost}}) \]

where \( k = \text{fastest}(M_e, \text{prices}, \max_{\mu}, \max_{\kappa}) \) \quad (3)

\[ \text{cheapest\_rate}(M_e, \text{prices}, \max_{\mu}, \max_{\kappa}, \text{rate}) = i \mid \min_{\forall i \in I} (i \mid \frac{\text{cost}(i, \text{prices})}{\text{cost}(k, \text{prices})} \leq \text{rate}, \prec_{\text{perf}}) \]

where \( k = \text{cheapest}(M_e, \text{prices}, \max_{\mu}, \max_{\kappa}) \) \quad (4)
Practical example

USE CASE (Experiment)

- Search using word:
  - `documents/search/tenant/D/replica/word`
- Return the last "number" documents:
  - `documents/tenant/D/tenant/replica/lim/number`
- Return the last documents:
  - `documents/tenant/D/replica/last`
- Return the document with the id 4:
  - `documents/tenant/D/replica/4`

Docker images:
- `mongo`
- `chumbo/arkiscrud:1.6.1`
### Practical example

\[ M_2 = \begin{pmatrix} 89.16 & 45.53 & 43.79 & 41.88 & 42.05 & 40.45 \\ 71.70 & 48.11 & 40.07 & 35.92 & 36.05 & 36.35 \end{pmatrix} \]

<table>
<thead>
<tr>
<th>Title</th>
<th>Policy</th>
<th>( \max_\mu )</th>
<th>( \max_\kappa )</th>
<th>Rate</th>
<th>#S-ful</th>
<th>#S-less</th>
<th>Cost</th>
<th>Makespan</th>
</tr>
</thead>
<tbody>
<tr>
<td>fastest</td>
<td>fastest</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>2</td>
<td>7</td>
<td>0.83</td>
<td>35.92</td>
</tr>
<tr>
<td>cheapest</td>
<td>cheapest</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>0.33</td>
<td>89.16</td>
</tr>
<tr>
<td>fastest with C</td>
<td>fastest</td>
<td>45.0</td>
<td>0.8</td>
<td>X</td>
<td>2</td>
<td>5</td>
<td>0.75</td>
<td>40.07</td>
</tr>
<tr>
<td>cheapest with C</td>
<td>cheapest</td>
<td>45.0</td>
<td>0.8</td>
<td>X</td>
<td>1</td>
<td>5</td>
<td>0.5</td>
<td>43.79</td>
</tr>
<tr>
<td>fastest(C &amp; rate)</td>
<td>fastest</td>
<td>45.0</td>
<td>0.8</td>
<td>1.06</td>
<td>1</td>
<td>7</td>
<td>0.58</td>
<td>41.88</td>
</tr>
<tr>
<td>cheapest(C &amp; rate)</td>
<td>cheapest</td>
<td>45.0</td>
<td>0.8</td>
<td>1.2</td>
<td>1</td>
<td>7</td>
<td>0.58</td>
<td>41.88</td>
</tr>
</tbody>
</table>

Legend: S-less = stateless, S-ful = stateful, C = constraints
Demo

DOCKER APP

\[ M_2 = \begin{pmatrix} 89.16 \times 45.53 \times 43.79 \times 41.88 \times 42.05 \times 40.45 \\ 71.70 \times 48.11 \times 40.07 \times 35.92 \times 36.05 \times 36.35 \end{pmatrix} \]

Formula script (Python)

<table>
<thead>
<tr>
<th>Title</th>
<th>Policy</th>
<th>( \max_\mu ), ( \max_\kappa ), Rate</th>
<th>#S-ful</th>
<th>#S-less</th>
<th>Cost</th>
<th>Makespan</th>
</tr>
</thead>
<tbody>
<tr>
<td>fastest</td>
<td>fastest</td>
<td>X X X</td>
<td>2</td>
<td>7</td>
<td>0.83</td>
<td>35.92</td>
</tr>
<tr>
<td>cheapest</td>
<td>cheapest</td>
<td>X X X</td>
<td>1</td>
<td>1</td>
<td>0.33</td>
<td>89.16</td>
</tr>
<tr>
<td>fastest with C</td>
<td>fastest</td>
<td>45.0 0.8 X</td>
<td>2</td>
<td>5</td>
<td>0.75</td>
<td>40.07</td>
</tr>
<tr>
<td>cheapest with C</td>
<td>cheapest</td>
<td>45.0 0.8 X</td>
<td>1</td>
<td>5</td>
<td>0.5</td>
<td>43.79</td>
</tr>
<tr>
<td>fastest (C &amp; rate)</td>
<td>fastest</td>
<td>45.0 0.8 1.06</td>
<td>1</td>
<td>7</td>
<td>0.58</td>
<td>41.88</td>
</tr>
<tr>
<td>cheapest (C &amp; rate)</td>
<td>cheapest</td>
<td>45.0 0.8 1.2</td>
<td>1</td>
<td>7</td>
<td>0.58</td>
<td>41.88</td>
</tr>
</tbody>
</table>

Legend: S-less = stateless, S-ful = stateful, C = constraints
Next steps

Until now:
- Specific workload

Future work:
- Formula to find the relation for each of the bi-directed connected graphs which compose the microservice architecture
Our research on Cloud-Native Applications

- One of the research initiatives of the Service Prototyping Lab at Zurich University of Applied Sciences

- Successful transformation of legacy software into cloud-native apps

- Proven track record with CRM, DMS and other Swiss business apps

We co-innovate with software SMEs - contact us for more information!
### Links

<table>
<thead>
<tr>
<th><strong>CNA research initiative</strong></th>
<th>blog.zhaw.ch/icclab/category/research-approach/themes/cloud-native-applications/</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Docker blog posts</strong></td>
<td><a href="https://blog.zhaw.ch/icclab/tag/docker/">https://blog.zhaw.ch/icclab/tag/docker/</a></td>
</tr>
<tr>
<td><strong>Application composed of multiple Docker containers</strong></td>
<td>github.com/serviceprototypinglab/scaling-containers</td>
</tr>
<tr>
<td><strong>Experiments - test</strong></td>
<td>github.com/serviceprototypinglab/scalability-experiments</td>
</tr>
<tr>
<td><strong>Results</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Open Science Notebook</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Formula</strong></td>
<td></td>
</tr>
</tbody>
</table>