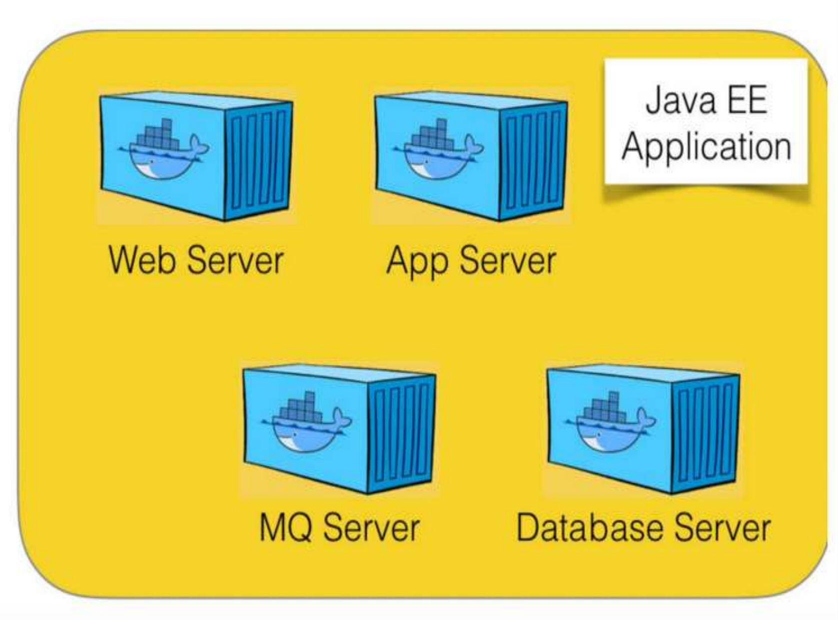
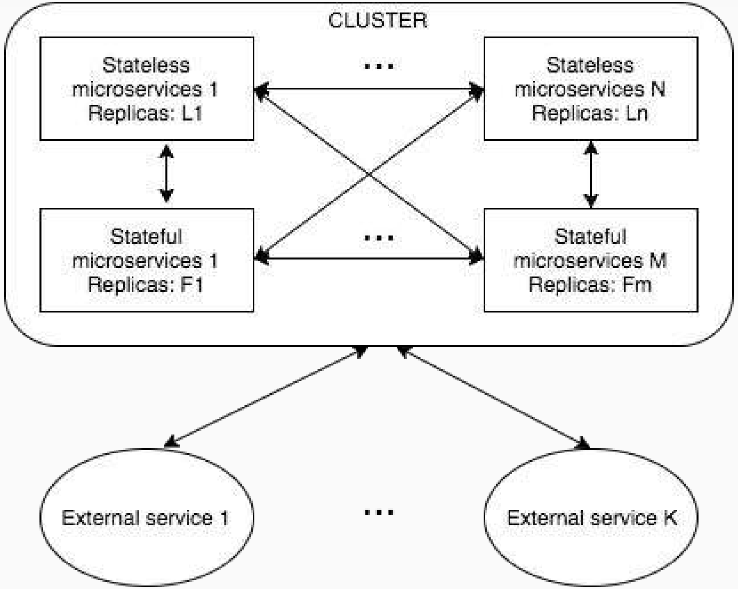
Predictable elasticity of Docker applications

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



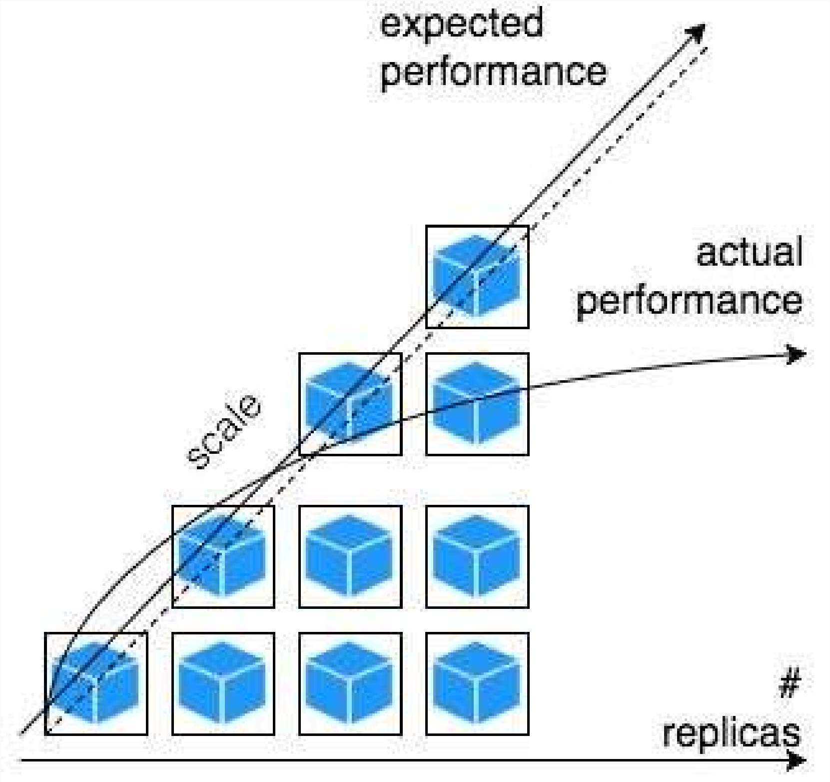
Example of docker app



Microservices architecture

ICCLAB SPLAB Service Engineering Research Area

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

Service Engineering Research Area
ICCLAB
SPLAB

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.

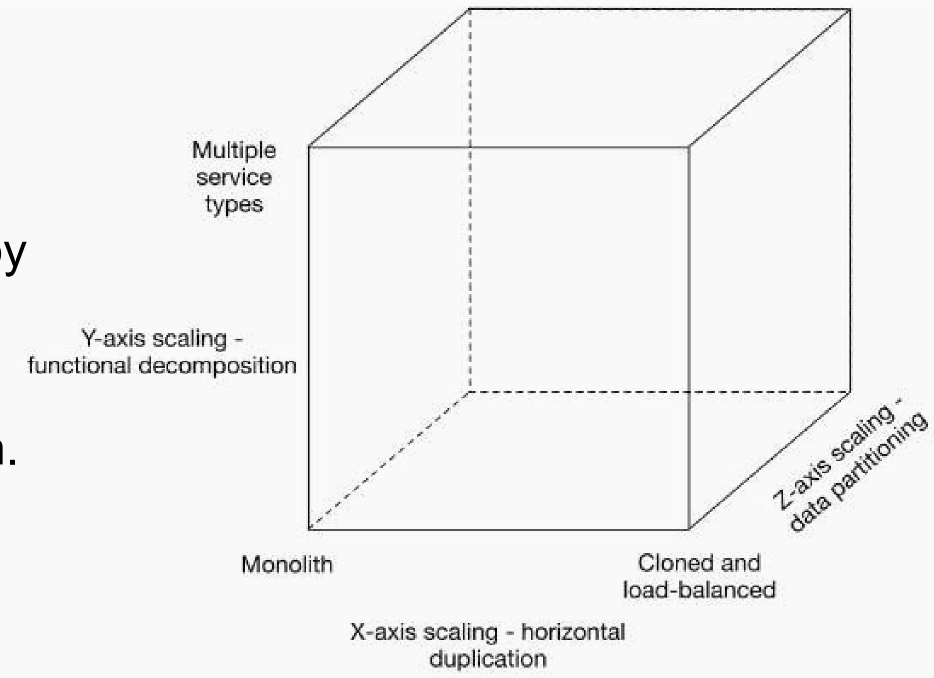
Stateless docker container: Do not keep any state and therefore do not persist data except through other services.

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.

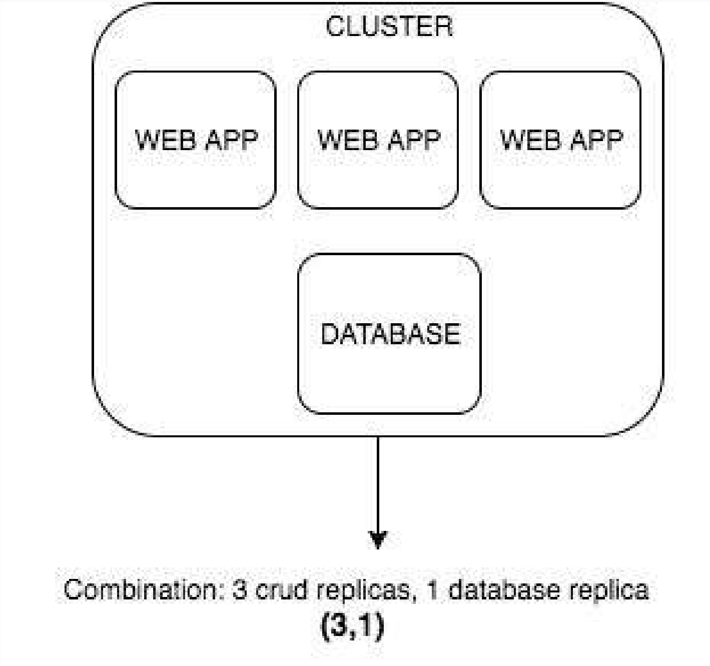
Some concepts

What is the best combination?

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

The 3 factors:

- Use case (fix)
- Cost
- Performance



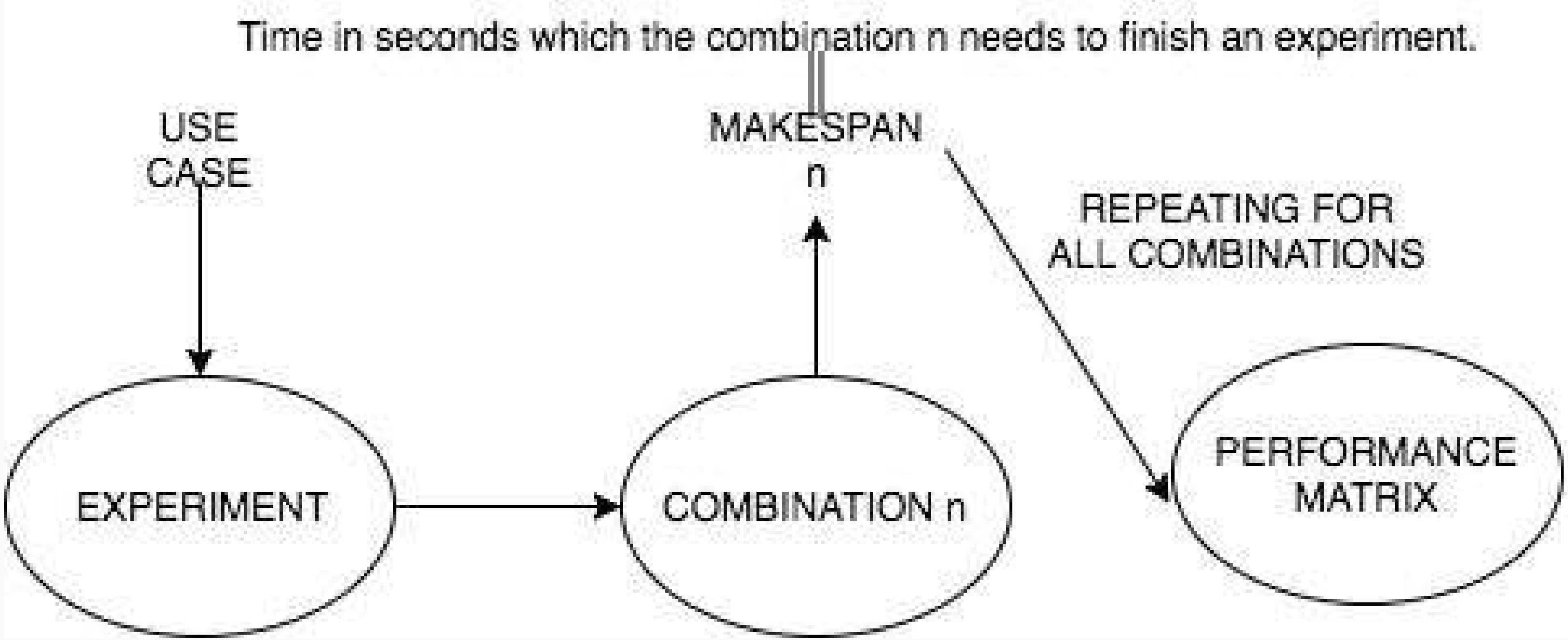
Example combination

Scalability formula

- What is the most economical combination satisfying minimum performance constraints?
- What is the fastest combination satisfying maximum price constraints?

ICCLAB SPLAB Service Engineering Research Area

Step 1: Create a performance matrix



And now some math to show how that works

Step 2: Obtain a solution

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

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

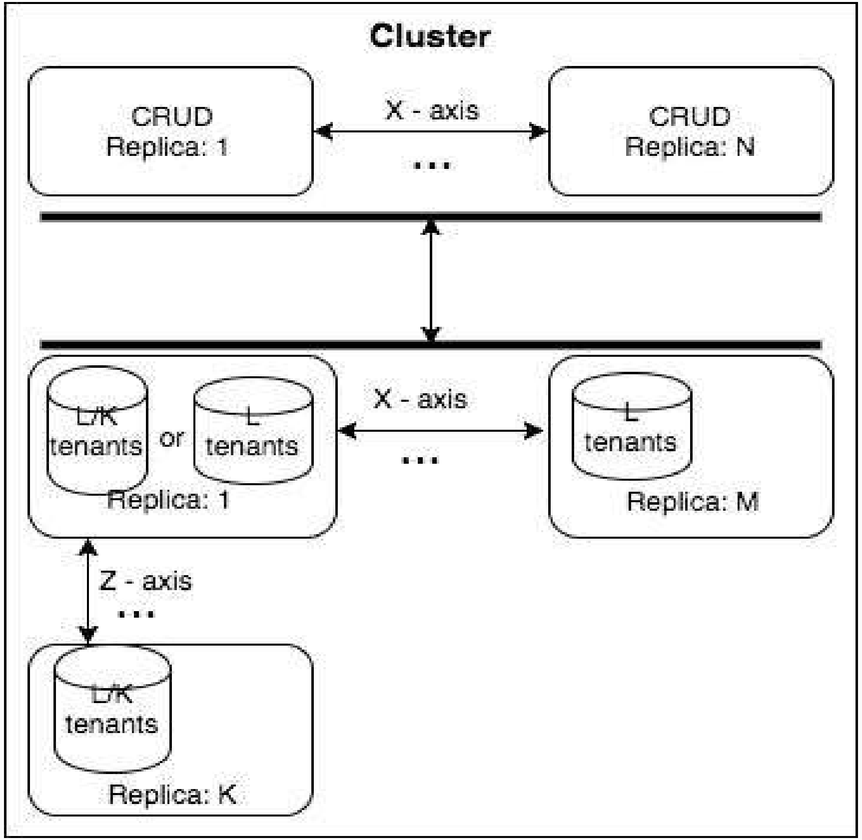
$$fastest\_rate(M_e, prices, max_\mu, max_\kappa, rate) = i \mid \min_{\forall i \in I}(i \mid \frac{m_i}{m_k} \leq rate, \prec_{cost})$$

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

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

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

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

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

Title	Policy	max_μ	max_κ	Rate	#S-ful	#S-less	Cost	Makespan
fastest	fastest	X	X	X	2	7	0.83	35.92
cheapest	cheapest	X	X	X	1	1	0.33	89.16
fastest with C	fastest	45.0	0.8	X	2	5	0.75	40.07
cheapest with C	cheapest	45.0	0.8	X	1	5	0.5	43.79
fastest(C & rate)	fastest	45.0	0.8	1.06	1	7	0.58	41.88
cheapest(C & rate)	cheapest	45.0	0.8	1.2	1	7	0.58	41.88
Legend: S-less = stateless, S-ful = stateful, C = constraints								

Demo

DOCKER APP

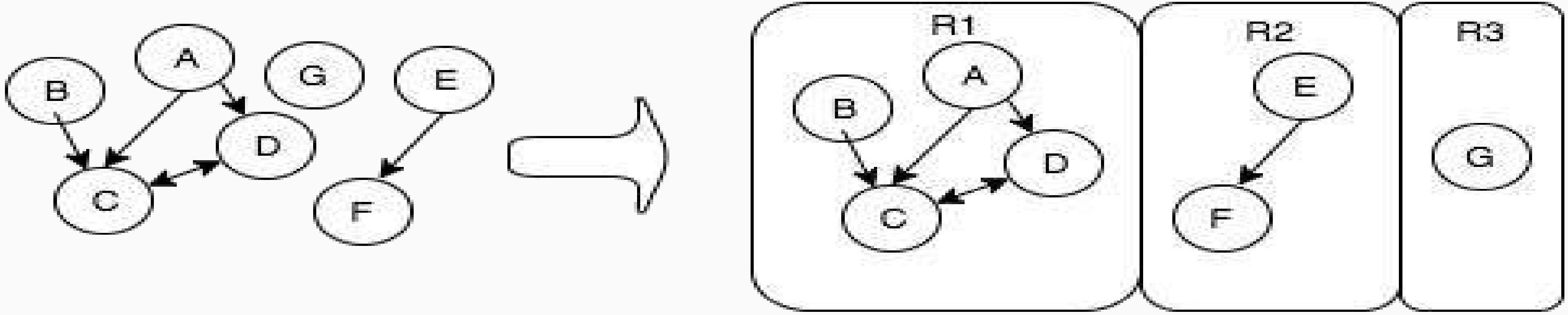
$$M_2 = \begin{pmatrix} 89.16 & * & 45.53 & * & 43.79 & * & 41.88 & * & 42.05 & * & 40.45 \\ 71.70 & * & 48.11 & * & 40.07 & * & 35.92 & * & 36.05 & * & 36.35 \end{pmatrix}$$

Formula script (Python)

Title	Policy	max_μ	max_κ	Rate	#S-ful	#S-less	Cost	Makespan
fastest	fastest	X	X	X	2	7	0.83	35.92
cheapest	cheapest	X	X	X	1	1	0.33	89.16
fastest with C	fastest	45.0	0.8	X	2	5	0.75	40.07
cheapest with C	cheapest	45.0	0.8	X	1	5	0.5	43.79
fastest(C & rate)	fastest	45.0	0.8	1.06	1	7	0.58	41.88
cheapest(C & rate)	cheapest	45.0	0.8	1.2	1	7	0.58	41.88

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

CNA research initiative	blog.zhaw.ch/icclab/category/research-approach/themes/cloud-native-applications/
Docker blog posts	https://blog.zhaw.ch/icclab/tag/docker/
Application composed of multiple Docker containers	github.com/serviceprototypinglab/scaling-containers
Experiments - test **Results** **Open Science Notebook** **Formula**	github.com/serviceprototypinglab/scalability-experiments

Open Cloud Day

Service Engineering Research Area
ICCLAB SPLAB