Towards Quantifiable Boundaries for Elastic Horizontal Scaling of Microservices

Manuel Ramírez López <ramz@zhaw.ch> & Josef Spillner <josef.spillner@zhaw.ch>
Service Prototyping Lab (blog.zhaw.ch/icclab)

Dec 5, 2017 | 6th CloudAM @ UCC, Austin, TX, USA
Motivation

Application...

... scaling:
  accommodate more users / growing workload
  desired: elasticity, rapidity

... auto-scaling:
  rule-based scaling actions
  trade-offs: effort for rule definition, initial calibration, hotspots

... pre-scaling (our work):
  determine initial combinatorial scaling
  fixed-workload vs. variable workload
Model

Microservices composition - three classes of services
Model

Scale cube (Abbott and Fisher, 2015)

Independently deployable microservices → Y axis (Hasselbring, 2016)
Assumptions

Application architecture following a microservice design
- stateful CRUD service
- replica count per service

Scenario implementation
- online document management application
- RESTful Python service, MongoDB

Scale cube relation
- X axis: horizontal replication
- Z axis: data partitions
Assumptions

Nonlinear constrained horizontal scaling behaviour on X axis according to following graph
Research Question and Approach

Question:

»Can the best combination of replicas for a given application and workload be calculated for performance-critical and cost-constrained settings?«

Approach:

• Formalisation of application structure, task, workload, environment + scaling constraints
• Combinations of scaling factors, optimal result vectors
Method: Optimality

What is the “best“ combination?

TG: Type graph
IG: Instance graph - replicas per microservice - 3x IG₀
Method: Formalisation

Mathematical model: m-dimensional makespan matrix

\[ M_e = M(e)_{n_1 \times \ldots \times n_m} = \begin{pmatrix} 
\mu_1,1,\ldots & \mu_1,2,\ldots & \ldots & \mu_1,n_2,\ldots \\
\mu_2,1,\ldots & \mu_2,2,\ldots & \ldots & \mu_2,n_2,\ldots \\
\ldots & \ldots & \ldots & \ldots \\
\mu n_1,1,\ldots & \mu n_1,2,\ldots & \ldots & \mu n_1,n_2,\ldots 
\end{pmatrix} \]

(2 out of m dimensions shown conforming scenario)

where:

- \( m \) - # of microservices
- \( n \) - # of replicas per microservice
  - stateful services: partitioning scheme (e.g. per tenant)
- \( e \) - experiment (task/workload combination)
- \( \mu \) - makespan
Method: Optimal Factors Formula

Three approaches
• unconstrained (baseline)
• constrained
• relaxed-constrained (with rate)

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

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

cost: resource cost or monetary cost
I: set of indices of M
Method: Complexity Reduction

Sparse matrices/arrays due to not fully connected microservices (TG level)
• representation: bi-directional disconnected graph
• vertices = microservices
• edges = connections (communication links)

Transformation: set of fully connected graphs

(caveat: not validated, relates to patterns - e.g. sidecar)
Implementation: Factor Injection

Integration with microservice management platforms
• e.g. container schedulers (Docker Compose, Kubernetes, ...)
• using placeholders in composition templates

Example as used in experiment:
Kubernetes 1.5 deployment @ Google Cloud Platform (GCP - GCE)

```json
{
    "kind": "Deployment",
    "apiVersion": "extensions/v1beta1",
    "metadata": {
        "name": "MICROSERVICE"
    },
    "spec": {
        "replicas": REPLICAS,
        "spec": {
            "containers": [
                {
                    "name": "MICROSERVICEIMPL",
                    "image": "NAMESPACE/CONTAINER:1.1",
                    ...
```
Implementation: Factor Injection

Verification through graphical user interface
Results

Stateless microservice: “arkisdocument“, API to search in documents
• from 1 to 11 replicas
Stateful microservice “mongodb“, 300 documents per tenant
• from 1 to 2 replicas
Workload generator/test microservice, not managed, not scaled

Cost/performance ratio is not linear.
Outlook: Variable Workloads

K experiments with maximum fulfilment of cost/performance requirements
Intersection analysis
Summary

Contributions
- formalised application scaling determination (X + Z axes in scale cube with microservice composition as Y axis)
- testbed based on Docker containers in Kubernetes
- practical use to complement autoscaling
- scientific open notebook for future work

https://github.com/serviceprototypinglab/scalability-experiments


«Microservices for Scalability» by Wilhelm Hasselbring, ICPE‘16 keynote