A Hybrid Approach to Microservices Load Balancing

Marco Autili, Alexander Perucci, and Lorenzo De Lauretis

Abstract During the past few years, microservices have been becoming a common architectural pattern increasingly used to realize flexible and scalable service-based applications. Microservices have grown in popularity as a mainstay in the business environment, allowing companies to increase development and maintenance speed, predict performance and scale, with scalability being one of the most important nonfunctional requirements to be fulfilled. Load balancing is the most prominent approach in support of scalability. In the realm of microservices, one usually distinguishes between two types of load balancers, namely, client-side and server-side load balancers. This work proposes a novel hybrid approach to microservices load balancing that combines the benefits of client-side and server-side load balancing.

1 Introduction

Microservices can be seen as a technique for developing software applications that, inheriting all the principles and concepts from the service-oriented architecture (SOA) style, permit to structure a service-based application as a collection of very small, loosely coupled software services. *Services are very small (micro) as for their contribution to the application, not because of their lines of code* [10]. In [18], James Lewis and Martin Fowler introduce microservices as:

> an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized

M. Autili (✉) · A. Perucci · L. De Lauretis
Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, L'Aquila, Italy
e-mail: marco.autili@univaq.it; alexander.perucci@univaq.it; lorenzo.delauretis@graduate.univaq.it

A. Bucchiarone et al. (eds.), *Microservices*,
https://doi.org/10.1007/978-3-030-31646-4_10

management of these services, which may be written in different programming languages and use different data storage technologies.

For what concerns the relationship between SOA and microservices, we can say that microservices are an architectural style competing with SOA in that they can be seen as (1) a synonym for "*SOA done right*" [29] and (2) an implementation approach to SOA [25]. In this sense, microservices can be seen as a substyle refining SOA with additional constraints [27].[1] Thus, as any SOA-based design technique, microservices describe a particular way of designing software applications as suites of independently deployable (micro)services, yet with stronger attention to isolation and autonomy. Microservices are independently scalable and can be replaced and upgraded independently, with the objective to support scalability. For this reason, microservices are becoming a common architectural pattern being increasingly used to realize flexible and scalable applications [1, 8, 27, 31, 34]. In particular, in the business environment, microservices have grown in popularity as a mainstay, allowing companies to increase their development and maintenance speed, predict performance and scale, with *scalability* being one of the most important nonfunctional requirements to be fulfilled.

Load balancing is the most prominent approach in support of scalability in a microservices architecture (MSA) [1, 4, 22]. The concept of load balancing spans many different application fields. Broadly speaking, in computing, a load balancer permits to distribute workloads across multiple computing resources, spanning from different server computers, through different network layers down to network links, from different processing units to disk drives. In a service-oriented setting, load balancing is the act of distributing service requests from clients to multiple servers that offer services, e.g., running in different containers distributed among physical or virtual machines. Containers allow the explicit specification of resource requirements, which makes it easy to distribute containers across different hosts so as to improve resource usage.

In the realm of microservices, load balancing concerns the arrival rates or concurrent number of requests [3]. The load balancer then attempts at distributing the workload of requests across multiple microservice instances with the aim to, e.g., optimize resource use, maximize throughput, minimize response time, and avoid bottlenecks (i.e., overload of any single instance). Moreover, characterizing and managing the workload of microservices is also beneficial in terms of cost for both cloud vendors and developers [33].

In the literature, there are many valuable works that, in different forms and for different purposes, concern load balancing. As also reported in [12, 30], a number of architectural patterns based on load balancing have also been proposed [7, 22, 23, 28, 35] (just to mention a few). Strictly focusing on microservices, state-of-the-art approaches, which are more closely related to our approach (Sect. 6), principally distinguish between two types of load balancing, namely, server-side and client-side

[1] A detailed analysis and comparison of microservices characteristics and principles can be found in [36].