Cloud Native Software Engineering Research Landscape

Brian Mitchell
College of Computing and Informatics
Drexel University, Philadelphia, PA, USA
bmitchell@drexel.edu

Abstract

Since the mid 2010's we have seen significant shifting of traditional brick-and-mortar businesses in adopting public cloud infrastructure. Early skepticism about moving strategic business to the cloud on based on security or technical risk have been replaced with aggressive plans to migrate that is well supported by both technical professionals and senior management. In fact, IDC estimates that the overall worldwide cloud spend will almost double from $706.6B in 2021 to $1.3T by 2025 [1].

It seems while everybody is embracing the cloud computing model, little attention has been paid to ways that mature software engineering practices needed to move to this new computing model at scale. Early successes by technology startups, or by industry moving their existing suite of web and mobile services over to the cloud are now being followed with ambitious plans to move more of their core business to the cloud, and to innovate new products and services that could only be delivered in the cloud.

While there is a significant research activity in many areas of cloud computing, little attention has been paid to the needs of the software engineering professional. This paper will focus on the landscape of Cloud Native Software Engineering from a practitioners standpoint, and identify some opportunities in this space that should be investigated by the research community.

1. Introduction

Delivering managed computing services via hosted infrastructure stared in the late 1990's with the introduction of the Software as a Service (SaaS) model. One of the earliest examples of this new model is Salesforce.com, that launched in 1999 [2]. Salesforce offered a best-in-class customer relationship management (CRM) suite, but its software was not as disruptive as its delivery model. Unlike other competitors that licensed you software that you ran on your own equipment, Salesforce pioneered the pay-as-you-go SaaS subscription model. In this model, you simply pay a monthly per-user charge and can access the solution from any device at any time.

While SaaS solutions , cloud computing as we know it today can be attributed to the launch of AWS (Amazon) [3] in early 2006, with Azure (Microsoft) [4] and GCP (Google) [5] following in 2008. Early adopters of cloud infrastructure were largely technology companies that innovated patterns, practices, and open sourced many tools and frameworks to highlight the benefits of running resilient and scalable business services in the public cloud.

2. What is Cloud Native Computing?

Before we explore the software engineering landscape for the cloud, we need to address exactly what we mean by cloud native computing. According to the Cloud Native Computing Foundation (CNCF) [6] *"Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds."*. Amazon's definition is *"Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds"*. Google offers the definition *"Cloud native means adapting to the many new possibilitiesbut very different set of architectural constraintsoffered by the cloud compared to traditional on-premises infrastructure."*. The primary theme in these definitions centers around the role that technologies play in enabling the creation of cloud native applications.

We think a better definition of cloud native computing that focuses more on software engineering is *"Cloud native applications are well architected systems, that are "container" packaged, and dynamically managed"*. Specifically:

- **Well Architected Systems** - By this we mean systems that adhere not only to established software engineering best practices but also embrace specific functional and non-functional capabilities offered by the cloud. For example, how are the computing components identified, how are they work with each other, how are security requirements met, how is the system designed for resiliency and scale?

- **Container Packaged** - The term *container* is overloaded in the cloud computing terminology landscape. In many places its equated to a standardized package that is managed by Docker [7] technologies - aka "a docker container". We take a more generic view of container packaging. Specifically, we think container packaging is a mechanism to package and deploy code that is ephemeral, can operate across a variety of different hardware architectures (e.g., Intel, ARM, etc), and at runtime is supervised. Supervision includes full lifecycle management associated with version identification, startup, shutdown, health checks, and monitoring. Examples of container packaging and supervision include Docker, Docker Compose, Kubernetes, and serverless [8]. We also include in this category the emerging popularity of using server-side web assembly [9, 10] as a way to package and deploy cloud native application services.

- **Dynamically Managed** - One interesting conceptual model for cloud computing is to consider the cloud as a large, highly distributed, special purpose operating system. Just like any operating system, there are a number of resource like storage, compute, network and security services that are needed by applications. The job of an operating system is to dynamically manage and optimize the allocation of these resources to the realtime computing demand on the system. When done well, every process being managed by the OS will perceive that it has access to the resources it needs, when it needs it. In a similar context, a cloud service provider, via Application Programming Interfaces (APIs), provides and manages resources to cloud native applications dynamically. What is generally different is that an OS manages physical resources on a system, whereas cloud resources are generally virtualized and distributed, and to a large degree fault-tolerant. For example, block storage that supports virtual machine reads and writes are automatically replicated across servers within an availability zone. Outside of initial configuration, the user does not worry about how durability is provided given its dynamically managed by the cloud service provider.

Now that we have provided a definition, there are a number of interesting software engineering problems that should be considered.

b v v v v
v vv
v vv vv v
v vv vv vvv v
v vv
v
v
vv
vv vv

3. Software Architecture Requirements

4. Software Construction Requirements

4.1. Polyglot Programming

References

[1] "IDC Forecasts Worldwide "Whole Cloud" Spending to Reach $1.3 Trillion by 2025". IDC. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS48208321

[2] "The History of Salesforce". Salesforce.com. [Online]. Available: https://www.salesforce.com/news/stories/the-history-of-salesforce/

[3] "About AWS". Amazon Web Services. [Online]. Available: https://aws.amazon.com/about-aws/

[4] "The History of Microsoft Azure". Microsoft. [Online]. Available: https://techcommunity.microsoft.com/t5/educator-developer-blog/the-history-of-microsoft-azure/ba-p/3574204

[5] B. Stevens. "Google Cloud Platform: your Next home in the cloud". [Online]. Available: https://cloud.google.com/blog/products/gcp/google-cloud-platform-your-next-home-in-the-cloud

[6] "Cloud Native Computing Foundation Homepage". CNCF. [Online]. Available: https://www.cncf.io/

[7] "Use containers to Build, Share and Run your applications", url = "https://www.docker.com/resources/what-container",. Docker.com.

[8] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski *et al.*, "Serverless computing: Current trends and open problems," in *Research advances in cloud computing*. Springer, 2017, pp. 1–20.

[9] A. Haas, A. Rossberg, D. L. Schuff, B. L. Titzer, M. Holman, D. Gohman, L. Wagner, A. Zakai, and J. Bastien, "Bringing the web up to speed with webassembly," in *Proceedings of the 38th ACM SIGPLAN Conference*

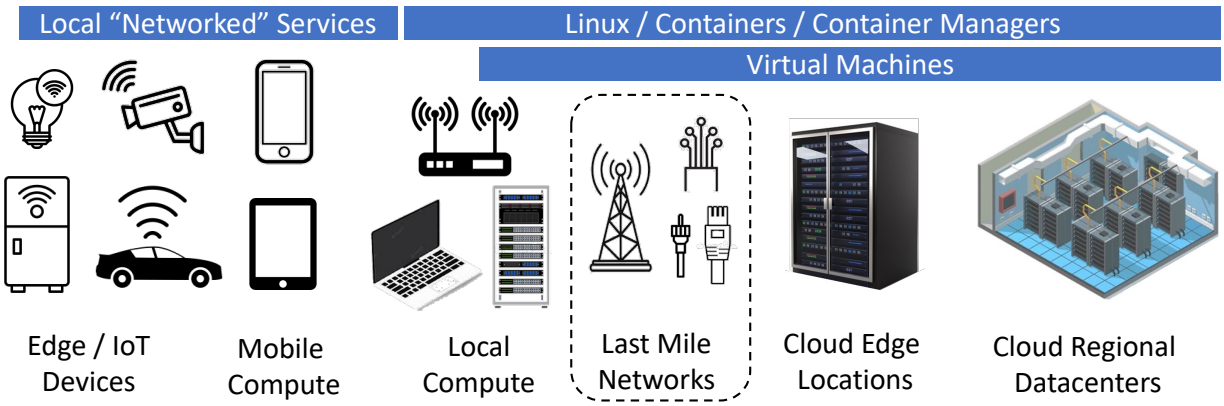

Figure 1. Cloud Compute - From Big to Small

on Programming Language Design and Implementation, 2017, pp. 185–200.

[10] B. Bosshard, "On the use of web assembly in a serverless context," in *Agile Processes in Software Engineering and Extreme Programming–Workshops*, 2020, p. 141.