2021 IEEE 26th International Conference on Emerging Technologies and Factory Automation (ETFA) | 978-1-7281-2989-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ETFA45728.2021.9613386

Cloud Native Computing for Industry 4.0: Challenges and Opportunities

Guilherme Gil[1][2], Daniel Corujo[1][2], Paulo Pedreiras[1][2]
[1]Department of Electronics, Telecommunications and Informatics, University of Aveiro, Aveiro, Portugal
[2]Instituto de Telecomunicações, Aveiro, Portugal

***Abstract*—Cloud-based architectures are advantageous in aspects such as scalability, reliability and resource utilization efficiency, to name just a few, thus being considered one of the pillars of Industry 4.0. However, in this domain, cloud computing platforms are subject to specific requirements, namely in what concerns real-time performance, determinism and fault-tolerance. This paper focuses on cloud native computing, which is an emerging and promising cloud-computing paradigm, specifically addressing its applicability to real-time systems. Firstly, it introduces the architecture of cloud native applications, discussing their principles, potential advantages and challenges. Then it addresses the opportunities and constraints of such technologies when applied to industrial real-time systems.**

***Index Terms*—Real-time systems, Industrial Informatics, Cloud native, Scheduling, Quality-of-Service**

I. Introduction

In recent years cloud computing has been considered one of the leading IT technologies. Together with Internet of Things (IoT) and Cyber-physical systems, it provides the means to digitalize industrial processes, being in the base of what is nowadays commonly designated as Industry 4.0. Even though the first cloud platform only appeared in 2002, with the launching of Amazon Web Services (AWS), the concept of cloud computing has its origins in 1960, when John McCarthy suggested that calculations would eventually be performed in public infrastructures. Nowadays, cloud computing is a widespread concept, with numerous companies actively using this technology to provide their services in an increasingly vast number of sectors [1].

One of cloud computing's main features is its computation capabilities [2], as it adopts efficient mechanisms to manage physical and virtual resources and respond to real-time events with parallel processing [3]. Furthermore, with the ongoing development of new and more efficient processors, data can now be processed faster and in larger volumes. However, due to it's heavy reliance on communication links to transmit information into or from the cloud [4], its rapid growth can instigate several issues if the underlying networks are not properly designed to manage the communication.

Over the last years, developers started to employ a new approach when creating and deploying applications in cloud environments, know as Cloud Native. With this model, applications are structured as a collection of several elements (e.g., microservices) that communicate with each other using lightweight event-based mechanisms [5]. However, despite being more efficient than standard cloud computing architectures, this approach still has the need for always-on components, requiring at least one container to be continuously running for each service.

In order to solve this issue, a cloud native development model known as Serverless computing, has started gaining popularity in the cloud community. According to this new model, developers only need to create stateless cloud functions that react to external events (e.g., HTTP requests) and deploy them in cloud platforms, thus providing an additional abstract layer. Despite making applications more flexible, Serverless computing prompts new challenges for cloud providers, namely regarding resource management and event scheduling for real-time applications. First, serverless workloads are extremely irregular, as function execution times can vary from several milliseconds to a few seconds [6]. Additionally, there is a substantial overhead that stems from container and function initialization, a phenomenon designated as ”cold starts” [7]. Current Serverless platforms such as AWS Lambda and Google Cloud functions do not include mechanisms to handle latency variations in workloads, which makes them unfitted for real-time applications [8].

Due to the success and wide acceptance of cloud computing, several authors already reviewed and studied this technology [9] [10], and specifically cloud resource management [11] [12]. Microservice based applications [13] [14] and containerization management was also addressed by others [15]. Nevertheless, there is still a lack of scientific contributions that review cloud native Serverless architectures that guarantee the requirements imposed by industrial real-time systems. Hence, this work provides a study of several cloud-based mechanisms that strive to provide such features, focusing specifically on resource management and scheduling.

The remainder of this document is structured as follows. Section II introduces the main cloud computing and cloud native concepts. Section III presents an overview of scientific contributions that address real-time features of cloud platforms and services, summarizing the most relevant challenges that must be addressed in order to allow the adoption of these technologies in the scope of industrial applications. Lastly, Section IV presents this work's conclusions.

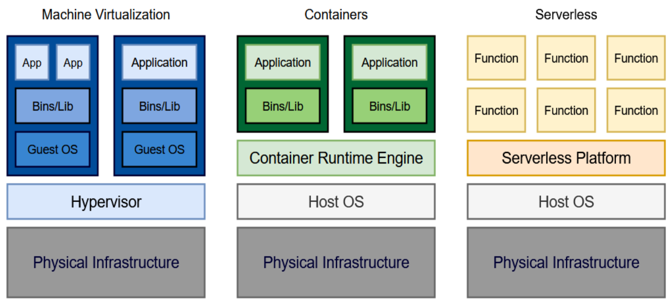

Fig. 1. Differences between the resource models of three cloud paradigms: a) virtual machines; b) microservices; c) Serverless, when developing cloud applications.

II. Background

This section briefly reviews basic concepts about cloud computing technologies, essential for the scope of this work.

A. Cloud Computing

Cloud computing is a technology where developers transfer their data and applications to the cloud, which can then be globally accessed through Internet services. According to the *NIST definition of Cloud computing*, IT resources are provided to consumers through three different service models [16]:

- **Infrastructure as a Service (IaaS)**: cloud providers supply physical and virtual IT resources (e.g., computing, storage and memory) to cloud consumers;
- **Platform as a Service (PaaS)**: cloud providers provide cloud platforms where consumers can deploy their applications and utilize ready-to-use services. Consumers cannot control the underlying infrastructure and nor its operating system;
- **Software as a Service (SaaS)**: offers applications running in cloud environments while hiding the underlying infrastructure. Consumers have no control over this infrastructure and most of the provided software (e.g., Office 365).

Figure 1 depicts an overview of different resource models used in cloud-based applications. Initially, these were deployed in bare metal server infrastructures. Machine virtualization eventually emerged, providing a more efficient means of handling physical resources. However, each virtual machine would still need an operating system image, adding memory overhead and increasing the overall system complexity. With the development of containerization and microservices, virtual machines were quickly dropped, as these new approaches were lighter and easier to manage. Finally, Serverless computing emerged, allowing developers to make better use of virtual resources. With a new scale-to-zero strategy, container units only execute if they have requests to process and only for the amount of time it takes to handle them [17].

B. Cloud native: Microservices vs Serverless computing

Generally, cloud-native applications comprise a set of components known as microservices. These microservices are deployed in self-contained deployment units (i.e., containers) [17]. This approach is more lightweight when compared to conventional virtual machines [14], however, it still had the issue that existed in standard cloud computing of always-on components while, at the same time, inciting additional challenges related to the deployment and scalability of each microservice. A new cloud-native model named Function-as-a-Service (FaaS), or Serverless computing, was proposed as a means to solve these problems. In [18] the authors state that, in Serverless computing, applications are deployed in event-triggered containers that may only activate once and are managed by a third party. While in SaaS, developers have no control over the cloud infrastructure, in FaaS, users have to deal with the logic of their applications using cloud functions.

Serverless applications rely on an event-based communication system, which forces deployment platforms to include mechanisms to manage and process these events as quickly as possible while guaranteeing scalability and fault tolerance. Serverless platforms may vary on the type of workflow employed. Usually, these comprise an event queue to store events, a dispatcher unit for scheduling and resource management, and workers (computational units where functions execute) (Figure 2). Regarding the events themselves, these can be, for example, HTTP requests, messages, emails, and so forth. Examples of current Serverless platforms include AWS lambda, OpenWhisk. Marbot is an example of a serverless application used by Slack to communicate with Amazon Web Services.

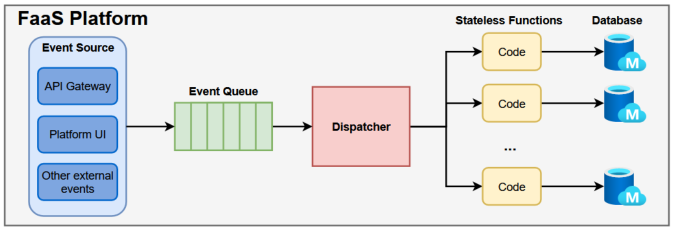

Fig. 2. General architecture for Serverless platforms (adapted from [14] [19] [20]).

Serverless platforms provide several benefits. From a consumer perspective, it reduces implementation and maintenance constraints as developers are now only responsible for creating the applications' code, thus shortening lead times. There are also fewer risks of resource misusage when scaling applications and higher flexibility with auto-scaling services [18]. However, no technology comes without its limitations, and Serverless computing is no exception. The main issues in Serverless platforms are related to latency constraints between inter-component communications and "cold starts". Moreover, there is also a lack of control over the deployed cloud infrastructure that is managed by third parties, security difficulties, testing complexity, and raises vendor lock-in situations [18].

III. Cloud technologies for real-time systems

With the introduction of Industry 4.0 and the corresponding massive digitalization of industrial processes, cloud computing provides the means for computational systems and sensors to be logically integrated, creating opportunities for new services, a concept known as Cloud of Things (CoT). However, in order to meet the requirements of industrial applications, these systems require complex architectures with specific mechanisms for resource provisioning and real-time computations and communications. Consequently, by taking advantage of the modular structure of cloud native applications, developers can now split their systems into several microservices/serverless functions, where each one can be updated and managed individually, thus providing flexibility in terms of scalability and resource provisioning.

According to Varghese and Tandur [21], Industry 4.0 comprises three main components: i) The application layer, responsible for all the automation processes that exist within the factory; ii) The network layer, which includes network control and data storage and processing in the cloud, and; iii) The physical layer, responsible for acquiring and computing data using sensors and actuators. Despite providing several benefits to manufacturers, Industry 4.0 also brings several challenges regarding its supporting infrastructure. For example, the authors describe the necessity of reliability and longevity in wireless communications for industrial environments. Other challenges involve dynamic resource provisioning and fast (i.e., low latency) communications. Cloud native approaches can tackle some of these issues and provide additional benefits to industrial environments. An important advantage of cloud native is the reliability and fault-tolerance provided by its architectures. The decomposition of the system into isolated elements results in errors being contained within a single component. Furthermore, element replacement/modification is cost-efficient and effortless, meaning that factories can easily adapt without being financially constrained. Lastly, with autoscaling mechanisms, cloud native architectures provide easier and faster means for manufacturers to scale their factories while only requiring the resources to maintain the applications.

A. Resource Management

Several works report different algorithms and architectures for cloud resource management and scheduling in order to provide the characteristics required to support real-time applications. For example, Mangla *et al.* [11] published a survey about resource scheduling in cloud environments. Their study was centered around four features: energy efficiency, virtual machine allocation, cost-effectiveness, and Service-Level Agreement (SLA). However, to achieve adequate resource provisioning in Serverless architectures, there are some additional issues to consider. Firstly, Serverless functions can have significant complexity differences, which vary their resource usage and compel applications to have specific resource schedulers. Then, there is also the matter of cold starts which causes severe latency increments due to the operations executed when a new function is invoked (i.e., container initialization, setting the run time environment, and function initialization). A feasible solution may involve the deployment of more containers than the assigned concurrency limit. This, however, leads to inappropriate usage of the available resources.

B. Scheduling in the cloud

Timeliness is one of the core dimensions of Quality-of-Service (QoS) in real-time systems. Providing suitable and guaranteed response times to critical tasks, even in peak load scenarios, is usually achieved through scheduling mechanisms that prioritize, in this case, certain functions or microservices. Even though some frameworks support the utilization of priority queues to handle events [22], these services are rather simplistic and can lead to lower priority packet losses and high latency. Thus, some authors proposed QoS-driven scheduling approaches for the cloud [23] [24].

Current Serverless platforms lack mechanisms to prioritize stateless functions, as most of them have fairly simple scheduling policies that do not guarantee service-level objectives (SLOs) [25]. Another issue, already mentioned, that prompts several scheduling challenges is cold starts. Because most platforms are reactive, they must wait for a new event to reuse a worker instance (warm container with function) or initiate a new one (cold container). This setup time may vary significantly, depending on the current workload, which highly impacts the scheduling and response-time of events. Other problems that stem from the lack of QoS guarantees in Serverless computing are [26]:

- **Incorrect concurrency limits:** Defined as the number of parallel functions operating at the same time, incorrect concurrency can stem from limits under/overestimate. It can lead to sudden request drops or resource misusage;
- **Mid-chain drops:** Occur whenever function invocations are not being handled. This results, for example, from exceeding concurrency limits, leading to request drops. It can lead to resources being wasted on incomplete function chains or other pressing issues when developers rely on function chains to complete a service.
- **Burst intolerance:** Bursts are specific workloads characterized by a continuous reception of requests in short time windows. If not correctly handled, they can lead to service delays and packet losses. This type of workload is hard to manage in Serverless platforms due to cold starts.

C. Discussion

The Cloud Native model is an effective approach to decompose complex software applications into smaller and more manageable parts. This design pattern allows structuring an application as a set of loosely coupled stateless services, combined with stateful backing services, being a promising architecture to handle the increasing complexity of industrial applications in the context of Industry 4.0.

Despite being widely used by the general public, Serverless platforms such as AWS Lambda and Microsoft Azure Functions are not designed to support real-time applications and require additional scheduling mechanisms. For instance, AWS Lambda only allows 100 parallel executions per account [27], which may not be enough to handle burst workloads. The scientific literature reports several attempts to address these limitations. Regarding resource management, Serverless platforms may be extended with a form of monitoring systems

[28], which provides real-time information about the current resource utilization and, based on that information, limit the utilization or allocate more resources, when required. In order to solve cold starts, a solution may involve maintaining warm containers to prevent invocation delays [29]. What concerns Serverless scheduling, the main objective is to provide better execution times while minimizing cost. This requires the adoption of a centralized or decentralized scheduler, together with specific scheduling algorithms for function prioritization. Several authors opted for the function's deadlines as a scheduling policy (e.g., [30]), thus ensuring the system timeliness.

IV. Conclusion

Cloud native is gaining momentum in cloud-based technologies. Its architecture, which fosters a design pattern that decomposes complex software applications into smaller and more manageable parts, seems particularly appealing to handle the increasing complexity brought by the Industry 4.0 paradigm. However, the roots of Serverless computing, a cloud-native model, are not in the industrial world, thus some aspects must be tackled to enable its use in this scope. Such challenges include cold start overheads, which can drastically increase the overall service latency, and unsuitable scheduling and communication mechanisms, which combined with a lack of suitable resource management schemes, don't allow attaining a deterministic application execution. The literature reports several contributions that address these issues, though partially, showing that, with due effort, this emerging paradigm may eventually be a solid base for the development of industrial applications, bringing advantages with respect to current approaches in terms of ease of development and deployment, maintenance, scalability, reliability and security, to name just a few. Future work includes carrying out a thorough characterization of existing platforms and develop a framework integrating methods to address the existing limitations, with the goal of obtaining a cloud native platform able to provide deterministic services.

Acknowledgement

This work has been supported by EC H2020 5GPPP 5Growth project (Grant 856709)

References

[1] O. Ali, A. Shrestha, J. Soar, and S. F. Wamba, "Cloud computing-enabled healthcare opportunities, issues, and applications: A systematic review," *International Journal of Information Management*, vol. 43, no. April, pp. 146–158, 2018.

[2] J. Lee, "A view of cloud computing," *International Journal of Networked and Distributed Computing*, vol. 1, no. 1, pp. 2–8, 2013.

[3] S. Marston, Z. Li, S. Bandyopadhyay, and A. Ghalsasi, "Cloud computing - The business perspective," in *Proceedings of the Annual Hawaii International Conference on System Sciences*, pp. 1–11, 2011.

[4] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of Network and Computer Applications*, vol. 98, pp. 27–42, 2017.

[5] J. Gilbert, *Cloud Native Development Patterns and Best Practices*. Packt Publishing, 2018.

[6] A. Singhvi, M. D. Shaikh, K. Houck, S. Venkataraman, A. Balasubramanian, and A. Akella, "Archipelago: A scalable low-latency serverless platform," *arXiv*, 2019.

[7] M. Stein, "The Serverless Scheduling Problem and NOAH," *arXiv*, 2018.

[8] H. D. Nguyen, C. Zhang, Z. Xiao, and A. A. Chien, "Real-time Serverless: Enabling application performance guarantees," in *WOSC 2019 - Proceedings of the 2019 5th International Workshop on Serverless Computing, Part of Middleware 2019*, pp. 1–6, 2019.

[9] L. Qian, Z. Luo, Y. Du, and L. Guo, "Cloud computing: An overview," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5931 LNCS, pp. 626–631, Springer, Berlin, Heidelberg, 2009.

[10] V. V. Arutyunov, "Cloud computing: Its history of development, modern state, and future considerations," *Scientific and Technical Information Processing*, vol. 39, no. 3, pp. 173–178, 2012.

[11] N. Mangla, M. Singh, and S. Rana, "Resource Scheduling in Cloud Environmet: a Survey," *Advances in Science and Technology Research Journal*, vol. 10, no. 30, pp. 38–50, 2016.

[12] R. Weingärtner, G. B. Bräscher, and C. B. Westphall, "Cloud resource management: A survey on forecasting and profiling models," *Journal of Network and Computer Applications*, vol. 47, pp. 99–106, 2015.

[13] D. Gannon, R. Barga, and N. Sundaresan, "Cloud-Native Applications," *IEEE Cloud Computing*, vol. 4, no. 5, pp. 16–21, 2017.

[14] N. Kratzke, "A brief history of cloud application architectures," *Applied Sciences (Switzerland)*, vol. 8, no. 8, pp. 1–26, 2018.

[15] J. Kosińska and K. Zieliński, "Autonomic Management Framework for Cloud-Native Applications," *Journal of Grid Computing*, vol. 18, no. 4, pp. 779–796, 2020.

[16] P. Mell and T. Grance, "The NIST-National Institute of Standars and Technology- Definition of Cloud Computing," *NIST Special Publication 800-145*, p. 7, 2011.

[17] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, "Cloud Container Technologies: a State-of-the-Art Review," *IEEE Transactions on Cloud Computing*, vol. 7, no. 3, pp. 677–692, 2017.

[18] M. Roberts and J. Chapin, "What is Serverless?." O'Reilly: Sebastopol, CA, USA, 2016.

[19] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski, and P. Suter, "Serverless computing: Current trends and open problems," *Research Advances in Cloud Computing*, pp. 1–20, 2017.

[20] M. Adhikari, T. Amgoth, and S. N. Srirama, "A survey on scheduling strategies for workflows in cloud environment and emerging trends," *ACM Computing Surveys*, vol. 52, no. 4, 2019.

[21] A. Varghese and D. Tandur, "Wireless requirements and challenges in Industry 4.0," in *Proceedings of 2014 International Conference on Contemporary Computing and Informatics, IC3I 2014*, pp. 634–638, Institute of Electrical and Electronics Engineers Inc., jan 2014. doi: 10.1109/IC3I.2014.7019732.

[22] "Implementing priority queueing with Amazon DynamoDB — AWS Database Blog." Oct. 16, 2019. Accessed on: Jan. 17, 2021. [Online]. Available: https://aws.amazon.com/pt/blogs/database/implementing-priority-queueing-with-amazon-dynamodb/.

[23] X. Wu, M. Deng, R. Zhang, B. Zeng, and S. Zhou, "A task scheduling algorithm based on QoS-driven in Cloud Computing," *Procedia Computer Science*, vol. 17, pp. 1162–1169, 2013.

[24] G. F. da Silva, F. Brasileiro, R. Lopes, F. Morais, M. Carvalho, and D. Turull, "QoS-driven scheduling in the cloud," *Journal of Internet Services and Applications*, vol. 11, no. 1, 2020.

[25] S. Shillaker, "A Provider-Friendly Serverless Framework for Latency-Critical Applications," in *EuroSys Doctoral Workshop (EuroDW)*, pp. 10–13, 2018.

[26] A. Tariq, A. Pahl, and E. Rozner, "Sequoia : Enabling Quality-of-Service in Serverless Computing," in *SoCC '20: Proceedings of the 11th ACM Symposium on Cloud Computing*, pp. 311–327, 2020.

[27] R. A. P. Rajan, "A review on serverless architectures-Function as a service (FaaS) in cloud computing," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 18, no. 1, pp. 530–537, 2020.

[28] Z. Li, Q. Chen, S. Xue, T. Ma, Y. Yang, Z. Song, and M. Guo, "Amoeba: QoS-Awareness and Reduced Resource Usage of Microservices with Serverless Computing," in *Proceedings - 2020 IEEE 34th International Parallel and Distributed Processing Symposium, IPDPS 2020*, pp. 399–408, 2020.

[29] E. Hunhoff, S. Irshad, V. Thurimella, A. Tariq, and E. Rozner, "Proactive serverless function resource management," *arXiv*, pp. 61–66, 2020.

[30] A. Das, A. Leaf, C. A. Varela, and S. Patterson, "Skedulix: Hybrid Cloud Scheduling for Cost-Efficient Execution of Serverless Applications," *arXiv*, no. iv, 2020.