A Survey on Techniques for Energy-Efficient Microservice-based Software Architectures in the Cloud

César Perdigão Batista, Sophie Chabridon, Denis Conan
(cesar-augusto.perdigao_batista, Sophie.Chabridon, Denis.Conan)@telecom-sudparis.eu
Samovar, Télécom SudParis, Institut Polytechnique de Paris
91120 Palaiseau, France

***Index Terms*—microservice, software architecture, energy efficiency, cloud**

Introduction

Microservices have emerged as a powerful architectural style in Cloud computing, impacting the way applications are developed, deployed, and managed. Unlike traditional monolithic architectures, microservices break down applications into smaller, independent components, each responsible for a specific function or task. This modular approach offers advantages in terms of scalability, flexibility, and resilience [1]. As a consequence, each microservice can be developed, tested, and instantiated independently, allowing teams to work concurrently and adopt diverse technologies and programming languages tailored to the specific needs of each service. In addition, the Cloud provides a suitable environment for microservice architectures, offering elastic infrastructure, managed services, and automation tools that facilitate the dynamic scaling and orchestration of these components.

In this work, we explore the energy efficiency of microservice-based software architectures in a Cloud computing environment. The outlined techniques are preliminary results of an ongoing Systematic Literature Review (SLR) which is in the conducting phase. It started with 345 studies for successive exclusions/selections and currently 66 studies are being analyzed.

While many energy-efficient techniques may benefit to various software architectures in the Cloud, certain techniques are particularly well-suited or even unique to microservices. The aim of this work is to identify promising software approaches to be applied to microservice applications, some typical Cloud technologies are also considered due to their symbiosis with the application layer. We selected some state-of-the-art techniques and present the main ideas relevant to energy-efficient microservice applications.

I. Application energy monitoring

Monitoring is the underlying approach to measure system's properties through specialized tooling and to quantify them. Thus, it is an essential part for the majority of energy-efficient techniques. There are several methodologies to monitor the energy consumption of resources in the Cloud [2]: *external devices* called wattmeters or powermeters usually provisioned in the wall socket of the equipment; *intra-resource* devices placed inside server nodes between the power supply and the motherboard; *hardware sensors and software interfaces* to report the consumption of an equipment, supporting monitoring tools via performance counters or vendor-specific APIs such as the Intel Running Average Power Limit (RAPL) interface or the NVIDIA Management Library (NVML) interface.

II. Dependency graph application modeling

In the context of microservices and containerized Cloud data centers, a dependency graph can model the interconnections and dependencies between different services, containers, and their resource requirements. This allows for a structured approach to understand, visualize, and optimize the deployment and orchestration of these services. In general, this model at run-time approach [3] aims to minimize operational expenses in Cloud data centers by optimizing container allocation, reducing communication workloads, and improving service performance.

III. Microservices Placement in Kubernetes

Container orchestration is a process that automates the deployment, management, scaling, and networking of containers across different environments. It effectively coordinates microservices to ensure they work together harmoniously. Kubernetes is the most popular container orchestration system, developed by Google and now open source. It manages communication, administration, and scheduling of containers in distributed clusters, including in the cloud. To deploy a container on a node, Kubernetes uses a scheduler that considers several factors to determine where to place a pod, including the availability of resources on the nodes, user-defined placement constraints, and fault tolerance policies. This ensures that resources are not overloaded and that applications run optimally.

IV. Workload prediction and management

Workload prediction consists of analyzing current and historical data to define a system profile on how the system behaves in the present and forecast future demands for its

resources. Predicting workload attributes helps to identify patterns and trends in resource usage, allowing system architects and operators to optimize their resource allocation strategies, ultimately reducing operational costs and energy [4].

V. Simulation and emulation of energy consumption

Simulations in Cloud computing refer to the use of computational models to replicate the behavior of Cloud systems under various conditions in a repeatable manner. These models can mimic the performance, scalability, and energy consumption of Cloud infrastructures, including server machines, networks, and storage. Reduced versions of real-world scenarios help researchers to predict how a Cloud system might behave without the complex requirements of provisioning, composition, configuration, and deployment of production environments [5]. However, simulators require assumptions and abstractions that may not precisely reflect the complexity of actual systems. To tackle this issue, emulation is employed to create an accurate replica of a system's hardware and software environment. While emulation offers higher fidelity, it is often more resource-intensive and less flexible compared to simulations [6].

VI. Configuration of soft resources

Unlike hardware resources (CPU, memory, network), soft resources are system software components, such as threads or connections, that are often neglected in resource allocation or optimization techniques [7]. Mainly, soft resources are system components that manage a server's concurrency level and enable the sharing of hardware resources.

VII. Application metadata for energy-awareness

Inclusion of software metadata is the practice of complementing the source code in order to provide additional information about the code that is not part of the program itself. It can be done in many forms, be it through annotations, configuration files, descriptors or expressions of a domain-specific language. For instance, energy-aware application annotations refer to metadata added to application components targeted to provide information regarding their energy consumption and efficiency characteristics. Also, in-code annotations that embed application requirements or behavior patterns can be useful for improving deployment strategies: e.g., by tagging microservices with energy-related metadata, infrastructures providers can gain insights into the energy usage patterns of different application components. This kind of metadata typically include details about the energy consumption profiles, energy-saving modes, mandatory and optional components, and the environmental impact of running the microservices under various conditions [8].

VIII. Energy consumption assessment at test time

Software energy consumption can be estimated by Energy Regression Testing (ERT) in Continuous Integration (CI) pipelines. Such type of test focuses on energy consumption of a program to ensure that code change do not degrade energy consumption [9]. In practice, when a code change is committed, a test from the CI pipeline would compare the two versions of the software and decide if the newer one would negatively impact the energy consumption. Particularly, ERT can break if a non-functional concern as the energy consumption is reported as increased or a target is reached, according to the tests' sensitivity configurations.

IX. Distributed tracing (DT) for energy-awareness

DT is a traditional technique for improving the visibility of requests into the operation of distributed systems. It is a type of correlated logging that helps debugging in production, investigating incidents, root cause analysis of failures, and performance profiling of an application across process, component, and machine boundaries [10]. There is a wide support of frameworks and tooling for DT in helping operators troubleshoot cross-component problems in deployed applications with rich and detailed diagnostic information [11]. Energy metrics, along with traditional system information (measurements and information about execution paths and graphs) can be included in the baggage context of the tracing [12].

Conclusion

Microservices enable to decouple applications into smaller, more manageable components that can be developed, tested, and deployed independently. This decoupling facilitates a low-granularity control of different parts of an application, in which development and operations can be optimized according to the usage requirements of the system. In situations in which energy efficiency is a priority, controlling a single or a small number of microservices can impact the energy consumption of the entire applications while balancing with performance requirements. To that end, the techniques reported in this work are promising tools to be leveraged towards reducing energy consumption of microservice-based applications in the Cloud.

Acknowledgments

This research was produced within the framework of Energy4Climate Interdisciplinary Center (E4C) of IP Paris and Ecole des Ponts ParisTech. This research was supported by 3rd Programme d'Investissements d'Avenir [ANR-18-EUR-0006-02].

This work received funding from the France 2030 program, managed by the French National Research Agency under grant agreement No. ANR-23-PECL-0003.

References

[1] C. Esposito, A. Castiglione, and K.-K. R. Choo, "Challenges in Delivering Software in the Cloud as Microservices," *IEEE Cloud Computing*, vol. 3, no. 5, pp. 10–14, 2016.

[2] F. Almeida, M. D. Assunção, J. Barbosa, V. Blanco, I. Brandic, G. Da Costa, M. F. Dolz, A. C. Elster, M. Jarus, H. D. Karatza *et al.*, "Energy monitoring as an essential building block towards sustainable ultrascale systems," *Sustainable Computing: Informatics and Systems*, vol. 17, pp. 27–42, 2018.

[3] G. Blair, N. Bencomo, and R. France, "Models@ run.time," *IEEE Computer*, pp. 22–27, Oct. 2009.

[4] B. Feng and Z. Ding, "GROUP: An End-to-end Multi-step-ahead Workload Prediction Approach Focusing on Workload Group Behavior," in *Proceedings of the ACM Web Conference*, ser. WWW '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 3098–3108. [Online]. Available: https://doi.org/10.1145/3543507.3583460

[5] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.995

[6] R. N. Calheiros, M. A. Netto, C. A. De Rose, and R. Buyya, "Emusim: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications," *Software: Practice and Experience*, vol. 43, no. 5, pp. 595–612, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2124

[7] J. Liu, Q. Wang, S. Zhang, L. Hu, and D. Da Silva, "Sora: A Latency Sensitive Approach for Microservice Soft Resource Adaptation," in *Proceedings of the 24th International Middleware Conference*, ser. Middleware '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 43–56. [Online]. Available: https://doi.org/10.1145/3590140.3592851

[8] D. M. Barbosa, R. G. De Moura Lima, P. H. Mendes Maia, and E. Costa, "Lotus@runtime: A tool for runtime monitoring and verification of self-adaptive systems," in *2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2017, pp. 24–30.

[9] B. Danglot, J.-R. Falleri, and R. Rouvoy, "Can We Spot Energy Regressions using Developers Tests?" *Empirical Software Engineering*, 2023. [Online]. Available: https://hal.science/hal-04286574

[10] A. Parker, D. Spoonhower, J. Mace, and R. Isaacs, *Distributed Tracing in Practice*. O'Reilly, 2020.

[11] T. Davidson, E. Wall, and J. Mace, "A Qualitative Interview Study of Distributed Tracing Visualisation: A Characterisation of Challenges and Opportunities," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 7, pp. 3828–3840, 2024.

[12] J. Mace and R. Fonseca, "Universal context propagation for distributed system instrumentation," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3190508.3190526