FREEDA: Failure-Resilient, Energy-aware, and Explainable Deployment of Microservice-based Applications over Cloud-IoT Infrastructures

Monica Vitali[3,†], Jacopo Soldani[1], Roberto Amadini[2,5], Antonio Brogi[1], Stefano Forti[1], Simone Gazza[2], Saverio Giallorenzo[2,4], Pierluigi Plebani[3], Francisco Ponce[1] and Gianluigi Zavattaro[2,4]

[1]*Department of Computer Science, University of Pisa, Italy*
[2]*Department of Computer Science and Engineering, University of Bologna, Italy*
[3]*Department of Computer Science, Electronic, and Bio-engineering, Politecnico di Milano, Italy*
[4]*INRIA, France*
[5]*OPTIMA ARC Industrial Transformation and Training Centre, Melbourne, Australia*

Abstract
FREEDA is an Italian research project aimed at supporting DevOps engineers in achieving failure-resilient and environmentally sustainable deployments of microservice-based applications over the Cloud-edge computing continuum. In this article, we describe motivations, objectives and first results of the FREEDA project, and discuss how FREEDA relates to the Information Systems Engineering community.

Keywords
Microservice-based Applications, Cloud-Edge, Application Deployment, Failure-resiliency, Energy-awareness, Sustainabile IT

Project overview Duration: September 2023 - September 2025. Consortium: University of Pisa (IT) (Project Coordinator), University of Bologna (IT), Politecnico di Milano (IT). Funding Agency: Ministry of Universities and Research (MUR), ITALY. Website: https://freeda.di.unipi.it

1. Project Overview

1.1. Context and Motivation

The extensive integration of smart connected devices, coupled with the increasing computational capacities they offer, requires a transformation of Cloud computing into ubiquitously

RPE@CAiSE'24: Research Projects Exhibition at the International Conference on Advanced Information Systems Engineering, June 3–7, 2024, Limassol, Cyprus
† Corresponding author.
monica.vitali@polimi.it (M. Vitali); jacopo.soldani@unipi.it (J. Soldani); roberto.amadini@unibo.it (R. Amadini); antonio.brogi@unipi.it (A. Brogi); stefano.forti@unipi.it (S. Forti); simone.gazza@unibo.it (S. Gazza); saverio.giallorenzo2@unibo.it (S. Giallorenzo); pierluigi.plebani@polimi.it (P. Plebani); francisco.ponce@di.unipi.it (F. Ponce); gianluigi.zavattaro@unibo.it (G. Zavattaro)
0000-0002-5258-1893 (M. Vitali); 0000-0002-2435-3543 (J. Soldani); 0000-0003-1668-7305 (R. Amadini); 0000-0003-2048-2468 (A. Brogi); 0000-0002-4159-8761 (S. Forti); 0000-0002-3658-6395 (S. Giallorenzo); 0000-0002-6411-0511 (F. Ponce)

distributed infrastructures that exploit computing capabilities at the edge of the network (Cloud-IoT) [1]. The distributed infrastructure on which the Cloud-IoT computing continuum is based is characterised by significant heterogeneity and variability. In such a scenario, the computing devices composing the infrastructure range from cloud servers to smart IoT devices. These nodes differ in terms of computational and storage capabilities, performance, cost, and ownership. In common scenarios as data-intensive applications, another aspect to be considered is the mutual position of the different nodes when they have to exchange information and the type of connection (i.e., wired or wireless) affecting the communication time and cost. At the same time, the infrastructure composition is continuously changing: nodes might join overtime or might become unavailable at some point, as well as their performance and capabilities might be affected by external factors (i.e., other applications concurrently running on the same hardware); additionally, the workload of the applications hosted in such an infrastructure is characterised by fluctuations in workload and traffic.

In this context, the widespread adoption of Microservice-based Applications (MSAs) in delivering enterprise solutions has increased the need for facilitating MSA deployment across the Cloud-IoT infrastructures seamlessly. An MSA is an application designed as a set of loosely coupled smaller components, each one with specific functionalities. Each component is also characterised by specific functional (e.g., amount of computational and storage resources needed) and non-functional (e.g., response time, latency) requirements that need to be considered and satisfied when the application is deployed on the infrastructure.

Given the complexity of both the Cloud-IoT infrastructure (i.e., number of nodes, heterogeneity, and variability) and the application (e.g., number of components and specific requirements), properly mapping each component to a feasible infrastructural node is becoming a complex and time-consuming task. This calls for approaches where the different deployment requirements of the microservices composing the application are considered together with the capabilities and features of the infrastructure. These approaches could influence the DevOps practice, which must take into consideration potential service and hosting node failures, as well as cascading failures where the malfunction of one infrastructure node or service leads to the failure of others [2].

The need for the deployment of resilient applications must also align with the recent European Union's requirements for sustainable IT [3][4], including the reduction of consumed energy. Even though cloud infrastructures' environmental impact has been significantly reduced in recent years, this reduction does not apply to edge devices or smaller data centers. Additionally, the deployed applications' energy demand has been increasing with the widespread adoption of the Cloud, due to the availability of large amounts of computational resources at a reduced cost. This is expected to occur in Cloud-IoT infrastructures as well, therefore calling for supporting energy-aware MSA deployments [5][6][7].

The combination of all these characteristics makes it difficult to reason on the Cloud-IoT deployment configuration of MSAs. Moreover, deployment requirements can conflict (e.g., resilience can be increased by deploying replicated service instances, which increases the toll on the energy budget). This calls for novel techniques and tools that help DevOps engineers to reason on deployment requirements.

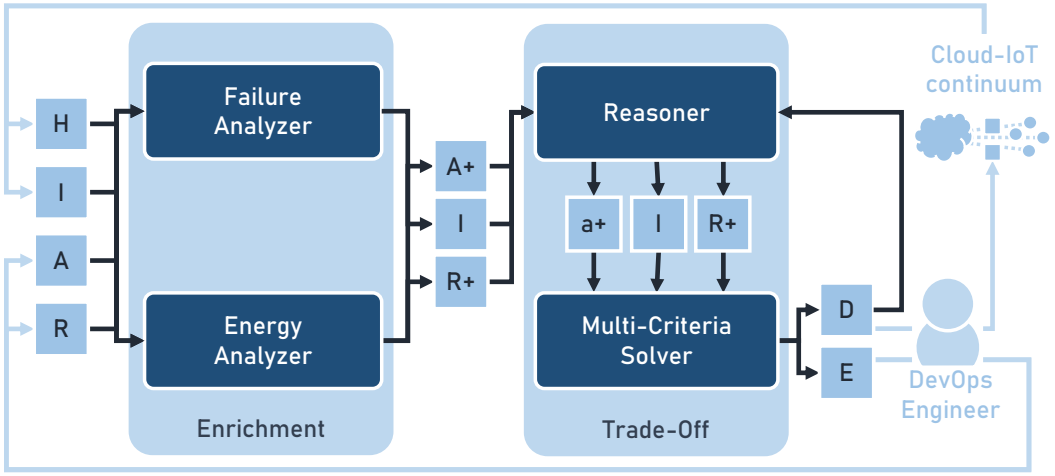

Figure 1: Overview of FREEDA's approach [8]

1.2. Proposed Approach

FREEDA aims to address the demand for DevOps support in deploying MSAs within Cloud-IoT infrastructures, integrating considerations for failure resilience and environmental sustainability. An overview of the proposed approach is illustrated in Fig. 1.

The project enables the holistic deployment of an MSA over a Cloud-IoT infrastructure by trading among its (possibly conflicting) deployment requirements. More precisely, FREEDA will enable analyzing an MSA and a Cloud-IoT infrastructure, together with information on the current and former MSA deployments (if any), to elicit additional requirements enabling the enforcement of the MSA deployment's failure resilience and energy sustainability.

As shown in Fig. 1, the proposed solution is divided into two main phases (*Enrichment* and *Trade-Off*) each one supported by two main components. The main stakeholder of the proposed approach is the DevOps Engineer willing to deploy an MSA in the infrastructure. The DevOps Engineer must provide the following information:

- A description of the application components (**A**): the set of microservices composing the application and their mutual relationships.
- A set of requirements (**R**): functional (computational and storage resources needed) and non-functional (performance, security, energy) requirements associated with each component.
- A description of the infrastructure (**I**): the set of nodes in which the application can be deployed and their capabilities and cost. These nodes might belong to the DevOps organization or an external organization (e.g., a public or private cloud provider).

Additionally, to these inputs provided by the DevOps engineer, we expect to have also information coming from the historical monitoring of the infrastructure and the application, if already deployed (**H**).

Using these inputs, FREEDA will provide the DevOps Engineer with a valid deployment able to fulfil all the expressed constraints. The core of the methodology is the **Multi-Criteria Solver** component. This component formally models the deployment problem, by encoding it in a constraint modelling language and builds a solver providing the best deployment plan (**D**).

Before being able to perform this step, the *Enrichment* phase needs to be executed. This phase aims at improving the application and requirements description to ensure the fulfilment of the failure resiliency and environmental sustainability requirements. The enrichment phase consists of two main components:

- **Failure Analyzer**: the focus of this component is to enrich the application and its requirements description with additional features. The failure identifies possible causes of failure in the application execution and includes proper sidecar integration components (e.g., circuit breakers) in **A** to avoid failure propagation. It also generates novel hard/soft requirements in **R** for enforcing failure resiliency at deployment time, e.g., avoiding deploying services on nodes that are known to fail if subject to a given load, or whose failure is predicted to occur soon based on the available historical data.
- **Energy Analyzer**: this component develops techniques to classify the profile of the nodes in a Cloud-IoT infrastructure and correspondingly generate requirements for reducing the energy consumed by MSAs deployed on such infrastructure. Also, the relation between an MSA's components, based on the connections/functional dependencies between microservices and information on the data they exchange are considered in the analysis to provide insights about microservice co-placement to the solver.

The output of the enrichment phase is an updated and enrichment description of the application components (**A+**) and requirements (**R+**), including failure-resiliency and energy-aware considerations. The enrichment phase reduces the design effort required from the DevOps Engineer by ensuring the satisfaction of these non-functional aspects in the deployment of the application without the need for her direct intervention.

As discussed in Sect. 1.1, FREEDA needs to operate in a dynamic environment that can affect both infrastructure and the application composition, as well as the applications' requirements. Anytime a significant change is detected or requirements get violated, a new execution of the **Multi-Criteria Solver** is needed to generate a new deployment plan. To avoid the inefficiency that might result from a full re-deployment of an already running application, the **Reasoner** component is in charge of determining a —possibly small— subset **a+** of the application components that need to be re-deployed due to violation of the requirements in **R+**. This continuous reasoning approach might generate a sub-optimal deployment plan while reducing the disruption of the re-deployment and ensuring the satisfaction of all the requirements.

The last relevant feature distinguishing the FREEDA approach from the state of the art is Explainability. FREEDA aims at providing DevOps engineers with proper and human-friendly explanations (**E**) on why/how specific deployment choices have been taken by the solver. Such explanations can also be exploited by the DevOps engineers to improve the MSA description or requirements. For instance, the explanation may suggest the deployment of additional components (e.g., circuit breakers) to reduce the risk of failure of another component, thus increasing the cost and energy consumption of the proposed solution. Based on this, a DevOps engineer may decide to refactor her service, natively including resiliency in the design.

2. Project Objectives and Expected Results

The project targets four main objectives:

(O_1) *Holistic MSA Deployment over the Cloud-Edge continuum*: FREEDA will develop novel techniques to determine a suitable trade-off among the MSA's deployment requirements, therein including cost, hardware, software, security, failure resilience, and sustainability requirements.

(O_2) *Continuous Reasoning for Adaptive MSA Deployment over the Cloud-Edge continuum*: FREEDA will develop continuous reasoning-like [9][10] to timely adapt the deployment of an MSA over a Cloud-Edge infrastructure when changes in the MSA, infrastructure, or deployment requirements occur.

(O_3) *Explainable Enhancement of MSA Deployments' Failure Resilience*: FREEDA will develop explainable techniques to enhance the failure resilience of to-be-deployed MSAs over Cloud-Edge infrastructures.

(O_4) *Explainable Reduction of MSA Deployments' Environmental Impact*: FREEDA will develop explainable techniques for reducing brown energy consumption when deploying MSAs over Cloud-Edge infrastructures.

FREEDA will deliver a set of techniques to fulfil the project's objectives, which will be prototyped to form a toolchain for planning MSA deployment across Cloud-IoT infrastructures. These techniques correspond to the main components outlined in Fig. 1. Development within FREEDA will adopt an iterative approach, beginning with basic implementations of proposed techniques and gradually refining and extending them until project goals are achieved. Each element of the approach will be provided as independent components, allowing DevOps Engineers to activate and configure them according to specific needs and objectives for their applications. The toolchain's components will be developed with a service-oriented approach. While the specific technologies and languages for implementation are still being examined, the aim remains to leverage standard and open-source solutions whenever possible. As an example, the deployment plan $\mathbf{D}$ and the descriptions of the application $\mathbf{A}$, requirements $\mathbf{R}$, and infrastructure $\mathbf{I}$ will be articulated in a YAML[1] file format, while the optimization problem will be formulated using MiniZinc [11] as constraint modeling language and solved with optimization tools supporting MiniZinc (e.g., Gurobi[2] or OR-Tools[3]).

The practical application of these techniques will be demonstrated through realistic use cases, illustrating the (re)configuration of MSA deployment over a testbed Cloud-IoT infrastructure. Operations and monitoring of the deployment will leverage existing tools, such as those based on TOSCA [12] or EDMM [13]. Additionally, FREEDA aims to engage with standardization committees (e.g., OASIS TOSCA) and open industrial initiatives (e.g., Cloud Native Computing Foundation, Gaia-X, Next Generation Internet, Industrial Internet Consortium). This dual approach will facilitate the exploration of emerging Cloud and Edge/Fog standards and solutions within FREEDA, while also contributing to standards and initiatives through the solutions developed within the project.

[1] https://yaml.org
[2] https://www.gurobi.com/
[3] https://developers.google.com/optimization

The prototype tools and their integration as a toolchain will be released as open-source software in public repositories, such as GitHub.

2.1. Preliminary Results

FREEDA has started its operations in September 2023. At the current stage, the partners have been involved in activities for the definition of:

- A preliminary version of the mathematical formalization of the optimization problem, which included a detailed description of the application, the infrastructure, and the requirements;
- A preliminary and simplified implementation of the solver using the model formalization to find a suitable deployment solution.

3. Relevance for the CAISE Community

The project's alignment with the themes of the International Conference on Advanced Information Systems Engineering (CAISE) underscores its relevance to the Information Systems Engineering community. Specifically, FREEDA addresses challenges that are of particular interest to this community, as delineated in the Call for Papers:

- Microservices design and deployment: this topic is aligned with the main motivation of the project as expressed in objective O1;
- Cloud- and edge-based IS engineering: this topic is aligned with the context in which FREEDA operates as declared in objectives O1 and O2;
- Context-aware, autonomous, and adaptive IS: this topic is aligned with the goal of the continuous reasoner component, adapting the deployment according to the context of execution and requirements violations as declared in objective O2.

Moreover, the project's focus resonates with key themes from previous editions of the conference. For instance, the Intelligent Information Systems theme from CAISE 2021 acknowledged the heightened level of uncertainty faced by organizations and the growing imperative for Intelligent Information Systems that offer trusted, adaptive, agile, and autonomous solutions. Similarly, the Resilient Information Systems theme from CAISE 2020 recognized the inherent complexity of information systems as they evolve, leading to susceptibility to various forms of degradation and failure. These topics are central to FREEDA's objectives.

Acknowledgments

This work was supported by the project FREEDA (CUP: I53D23003550006), funded by the frameworks PRIN (MUR, Italy) and Next Generation EU.

References

[1] A. J. Ferrer, J. M. Marquès, J. Jorba, Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing, ACM Computing Surveys (CSUR) 51 (2019) 1–36.

[2] J. Soldani, A. Brogi, Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: A survey, ACM Computing Surveys (CSUR) 55 (2022) 1–39.

[3] EU, A new strategic agenda 2019 – 2024, https://www.consilium.europa.eu/media/39914/a-new-strategic-agenda-2019-2024.pdf, 2019. Accessed: 2024-04-12.

[4] EU, Eu. financing the climate transition, https://www.consilium.europa.eu/en/policies/climate-finance/, 2021. Accessed: 2024-04-12.

[5] S. Forti, A. Brogi, Green application placement in the cloud-iot continuum, in: International Symposium on Practical Aspects of Declarative Languages, Springer, 2022, pp. 208–217.

[6] M. Vitali, Towards greener applications: Enabling sustainable-aware cloud native applications design, in: International Conference on Advanced Information Systems Engineering, Springer, 2022, pp. 93–108.

[7] M. Vitali, P. Schmiedmayer, V. Bootz, Enriching cloud-native applications with sustainability features, in: 2023 IEEE International Conference on Cloud Engineering (IC2E), IEEE, 2023, pp. 21–31.

[8] J. Soldani, R. Amadini, A. Brogi, et al., Towards Sustainable Deployment of Microservices over the Cloud-Edge Continuum, with FREEDA, in: International Workshop on Flexible Resource and Application Management on the Edge (FRAME), ACM, 2024, pp. 1–4.

[9] S. Forti, et al., Declarative continuous reasoning in the cloud-iot continuum, J. Log. Comput. 32 (2022) 206–232. doi:10.1093/LOGCOM/EXAB083.

[10] P. O'Hearn, Continuous reasoning: Scaling the impact of formal methods, in: LICS 2018, ACM, 2018, p. 13–25. doi:10.1145/3209108.3209109.

[11] N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, G. Tack, MiniZinc: Towards a standard CP modelling language, in: International Conference on Principles and Practice of Constraint Programming, Springer, 2007, pp. 529–543.

[12] M. Rutkowski, C. Lauwers, C. Noshpitz, C. Curescu (eds.), TOSCA Simple Profile in YAML Version 1.3, https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/os/TOSCA-Simple-Profile-YAML-v1.3-os.html, 2020. Accessed: 2024-04-12.

[13] M. Wurster, U. Breitenbücher, A. Brogi, F. Diez, F. Leymann, J. Soldani, K. Wild, Automating the deployment of distributed applications by combining multiple deployment technologies., in: CLOSER, 2021, pp. 178–189.