GoSpeechLess: Interoperable Serverless ML-based Cloud Services

Sashko Ristov
sashko.ristov@uibk.ac.at
University of Innsbruck
Innsbruck, Tyrol, Austria

Philipp Gritsch
philipp.gritsch@uibk.ac.at
University of Innsbruck
Innsbruck, Tyrol, Austria

David Meyer
david.meyer@student.uibk.ac.at
University of Innsbruck
Innsbruck, Tyrol, Austria

Michael Felderer
michael.felderer@dlr.de
German Aerospace Center (DLR)
Cologne, Germany

ABSTRACT

Recently, Backend-as-a-Service (BaaS)-enabled serverless functions have been rapidly gaining traction. However, the dependence on specific provider features and configurations still leads to challenges in terms of portability, underlying platform heterogeneity, and vendor lock-in. To bridge this gap, this poster introduces *GoSpeechLess*[1], a GoLang-library for serverless functions that allows developers to code portable functions with interoperable BaaS services in a uniform manner. *GoSpeechLess* thereby is able to reduce development effort by improving maintainability index by up to 40 % and reducing LOC by up to 75 %.

CCS CONCEPTS

• **Computer systems organization → Cloud computing**; • **Software and its engineering → Interoperability**.

ACM Reference Format:
Sashko Ristov, Philipp Gritsch, David Meyer, and Michael Felderer. 2024. GoSpeechLess: Interoperable Serverless ML-based Cloud Services. In *Proceedings of 46th International Conference on Software Engineering (ICSE 2024).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Function-as-a-Service (FaaS) is an emerging programming paradigm in cloud computing which simplifies the management and administration of cloud computing resources because developers can simply publish their code in the form of functions while the infrastructure management is delegated to the cloud providers. With serverless computing being increasingly applied in a wider set of use cases, the need to create more complex functions which can interact with external services [2] naturally arises. We refer to such functions as *BaaS-enabled FaaS functions* because they typically rely on auxiliary cloud services (BaaS) such as AWS S3, GCP Text-to-Speech, etc., to perform those tasks. Developers deploy their serverless applications across multiple cloud providers to reduce cost [1], or improve performance with higher scalability [4].

However, current serverless applications are still largely dependent on the specific cloud provider service offerings and implementations. Porting serverless applications across clouds requires development and integration effort that is only partially diminished by current state-of-the-art approaches. Serverless programming models address portability by offering abstractions at the function level but would require additional development effort also to address intra-function portability. Hence, dealing with underlying platform heterogeneity and inherent vendor lock-in requires addressing many serverless-specific issues. For one, services of different providers comprise different feature sets, like volume or audio encoding for services that offer Text-To-Speech conversion. For another, different providers offer different SDKs, requiring developers to either read the documentation of every used provider or abstain from using the services of other providers.

To bridge these gaps in state-of-the-art approaches for the realistic use cases "*serverless = FaaS + BaaS*" [3] in federated Faas, this poster introduces *GoSpeechLess*, a multi-cloud library that allows developers to code portable *BaaS-enabled functions* with portable and dynamic Text-To-Speech BaaS service implementations. *GoSpeechLess* allows developers to write serverless functions without being tied to the specific SDKs of various cloud (and BaaS) providers. The library provides portability and interoperability by dynamically invoking different BaaS implementations with a single API call. Its programming abstractions simplify the development process and open new doors for flexibility and efficiency in deploying and managing serverless applications.

2 GOSPEECHLESS APPROACH

Figure 1 shows how *GoSpeechLess* performs syntactic input adaptations from the common input format of the service into the specific input syntax of the selected BaaS service implementation. Moreover, different BaaS service implementations also differ in semantic requirements. For example, GCP's Text-To-Speech implementation supports volume configuration, while AWS's implementation supports adjusting the volume by providing and modifying the Speech-Synthesis-Markup-Language (SSML). Our library opaquely translates the input to the provider-specific semantics.

When all adaptations of the inputs are done, *GoSpeechLess* uses the specific SDK of the selected provider to invoke the concrete

[1] https://github.com/FaaSTools/GoText2Speech

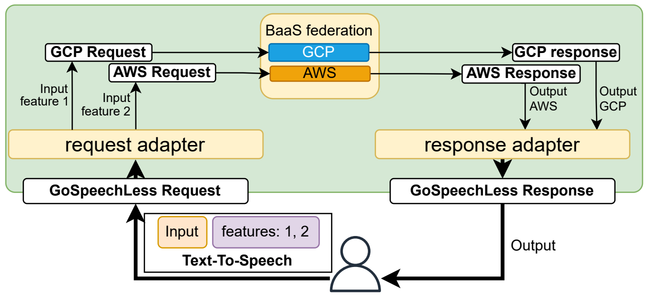

Figure 1: *GoSpeechLess* high-level approach.

implementation of the BaaS service. Our library calls the BaaS service with the adapted input for the BaaS service of the already selected provider, additionally regarding the supported format of the input data. If a requested feature (e.g., a specific audio encoding) isn't available for one provider, *GoSpeechLess* will automatically invoke the BaaS service implementation that supports the requested feature. Analog to reformatting the syntax and semantics of the input, *GoSpeechLess* adapts the output of the selected BaaS service implementation into the commonly defined output format, which is exposed to the developer.

GoSpeechLess supports the developer to specify the location where the input and result of the BaaS service should be stored. If this option is used, *GoSpeechLess* stores the output on the storage in a transparent way, regardless of the location and the provider of the selected BaaS service, the function that invokes it, and the location of the input and output data.

3 RESULTS AND DISCUSSION

We implemented *GoSpeechLess* library in GoLang, offering developers a Text-To-Speech BaaS service abstraction. The library currently supports AWS Polly and GCP Speech Synthesis. To maximize the coverage of our evaluation, we considered various metrics to evaluate *GoSpeechLess*. We evaluated *GoSpeechLess* with different input and output data locations and required features. For this purpose, we evaluated the following four metrics: *LoC*, *maintainability index*[2], and *overhead*.

Our observations indicate that *GoSpeechLess* requires significantly less LoC for all providers. It reduces LoC between 56.7 % and 75.4 % for AWS and from 59.4 % and up to 73.9 % for GCP.

The maintainability index is increased by 17.86 % for AWS and 17.59 % for GCP. Notably is the improvement of the maintainability index by 23.19 % for the configuration in which the function accesses the storage for both input and output. Moreover, 4 out of 6 native implementations are difficult to maintain, while all *GoSpeechLess*'s implementations are moderately maintainable.

Cyclomatic complexity is reduced from 8.7 to 5 for AWS (42.31 %) and from 8 to 3 for GCP functions (62.5 %). The Halstead volume is also reduced from 1,676.5 to 1,061.5 for AWS (36.68 %) and from 1,755.6 to 1,051.1 for GCP functions (40.13 %).

Since *GoSpeechLess* integrates dependencies of both providers, we evaluate the overhead that *GoSpeechLess* causes to functions that use the library. We observe that, in general, *GoSpeechLess* increases the runtime. The functions using *GoSpeechLess* reported an overhead of 9.21 % and 1.13 % on AWS and GCP, respectively.

GoSpeechLess's *Service abstraction* enables applications to transparently and interoperably use different BaaS service implementations without requiring additional development. *GoSpeechLess* is designed to simplify the extension of new BaaS service implementations, that is, to add a BaaS service implementation of a new provider, such as Azure, IBM, or Alibaba. Furthermore, if a provider changes their SDK, only the *GoSpeechLess* code needs to be updated, without requiring rewriting of any function that already used the defined interfaces.

4 CONCLUSION AND FUTURE WORK

We introduced *GoSpeechLess*, a novel library that significantly reduces development efforts to code serverless functions that federate BaaS services of various providers. The current implementation of *GoSpeechLess* supports Text-To-Speech BaaS services from two providers, AWS and GCP.

To the best of our knowledge, *GoSpeechLess* is the first framework that exposes common APIs to the developer that hide the heterogeneity of SDKs for BaaS services, providing fully interoperable and portable serverless applications using BaaS services. Additionally, the library integrates the features that are supported by different providers and transparently calls the fitting BaaS service implementation. We believe that it will be used by developers, and we invite the research community to join in its extension for other providers and BaaS services. We will publish all follow-up extensions on the same GitHub organization[3].

GoSpeechLess significantly reduces development effort in terms of LoC from 56.7 % compared to coding a single SDK, by up to 75.4 %. It also increases the maintainability index (Halstead volume) by up to 40.13 %. The trade-off is an imposed runtime overhead up to 9.21 %.

We will extend our work in several directions. First, we will analyze the differences between other BaaS services and add support in the GoSpeechLess library. Secondly, we will integrate support for the same BaaS services of other providers. Finally, we will implement GoSpeechLess in other programming languages, such as Python, Node.js and Java, which are widely used to code functions.

REFERENCES

[1] Juan J. Durillo, Radu Prodan, and Jorge G. Barbosa. 2015. Pareto tradeoff scheduling of workflows on federated commercial Clouds. *Simulation Modelling Practice and Theory* 58 (2015), 95–111. https://doi.org/10.1016/j.simpat.2015.07.001 Special Issue on TECHNIQUES AND APPLICATIONS FOR SUSTAINABLE ULTRASCALE COMPUTING SYSTEMS.

[2] Simon Eismann, Joel Scheuner, Erwin van Eyk, Maximilian Schwinger, Johannes Grohmann, Nikolas Herbst, Cristina L. Abad, and Alexandru Iosup. 2022. The State of Serverless Applications: Collection, Characterization, and Community Consensus. *IEEE Transactions on Software Engineering* 48, 10 (2022), 4152–4166. https://doi.org/10.1109/TSE.2021.3113940

[3] Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-Che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Yadwadkar, et al. 2019. Cloud programming simplified: A berkeley view on serverless computing. *arXiv preprint arXiv:1902.03383* (2019).

[4] Sasko Ristov, Stefan Pedratscher, and Thomas Fahringer. 2023. xAFCL: Run Scalable Function Choreographies Across Multiple FaaS Systems. *IEEE Transactions on Services Computing* 16, 1 (2023), 711–723. https://doi.org/10.1109/TSC.2021.3128137

[2]https://github.com/yagipy/maintidx

[3]https://github.com/FaaSTools