Poster: Function Delivery Network: Extending Serverless to Heterogeneous Computing

Anshul Jindal*, Mohak Chadha*, Michael Gerndt*, Julian Frielinghaus*, Vladimir Podolskiy*, Pengfei Chen†,
*Chair of Computer Architecture and Parallel Systems, Technische Universität München, Garching (near Munich), Germany
*Email: {anshul.jindal, mohak.chadha}@tum.de, gerndt@in.tum.de, {julian.frielinghaus, v.podolskiy}@tum.de
†School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China
†Email: chenpf7@mail.sysu.edu.cn

***Abstract*—Several of today's cloud applications are spread over heterogeneous connected computing resources and are highly dynamic in their structure and resource requirements. However, serverless computing and Function-as-a-Service (FaaS) platforms are limited to homogeneous clusters and homogeneous functions.**

We introduce an extension of FaaS to heterogeneous computing and to support heterogeneous functions through a network of distributed heterogeneous target platforms called Function Delivery Network (FDN). A target platform is a combination of a cluster of a homogeneous computing system and a FaaS platform on top of it. FDN provides Function-Delivery-as-a-Service (FDaaS), delivering the function invocations to the right target platform. We showcase the opportunities such as collaborative execution between multiple target platforms and varied target platform's characteristics that the FDN offers in fulfilling two objectives: Service Level Objective (SLO) requirements and energy efficiency when scheduling functions invocations by evaluating over five distributed target platforms.

***Index Terms*—Serverless, Function-as-a-Service, Heterogeneous Computing, Function Delivery Network**

I. Introduction

Programming and deploying the applications that are spread over the computing continuum is highly challenging due to the heterogeneity of the underlying hardware, varying compute and data access requirements across time and application components. Serverless computing has made the deployment of applications and their scaling easier by abstracting the server management and infrastructure decisions away from the users [1]. However, Function-as-a-Service (FaaS), a key enabler of serverless computing is limited to homogeneous computing as well as to homogeneous functions. Furthermore, FaaS platforms do not account for the underlying heterogeneity in the system architectures. As a result, application developers cannot optimize the applications functions for various heterogeneous systems like mini-computers (such as edge devices), High Performance Computing nodes, and FPGAs.

In this work, we introduce an extension to the concept of FaaS as a programming interface for heterogeneous computing and to support heterogeneous functions with varying computational and data requirements. This extension is a network of distributed heterogeneous target platforms called Function Delivery Network (FDN) analogous to Content Delivery Networks [2]. A target platform is a combination of a cluster of homogeneous nodes and a FaaS platform on top of it. FDN provides Function Delivery as a Service (FDaaS), delivering the function to the right target platform based on the required computational and data demand.

The automatic management of resources in the proposed serverless based FDN facilitates application development by shifting the deployment burden of the application function for the right target platform to cloud platform. The heterogeneity of the resources in the continuum is specifically challenging for resource management and the integration of systems like IoT Greengrass from Amazon [3] with application lambda functions deployed to the edge and ARM-based SoCs with the hardware programmability of an FPGA like Zynq boards for general FaaS applications and will require an extension of the FaaS platforms across heterogeneous devices. However, due to the heterogeneity of the FDN, it offers a wide range of opportunities for meeting different objectives like SLO requirements and energy efficiency in unconventional ways. Towards this, we show based on our experiments the opportunities offered by FDN in meeting these two objectives.

II. Function Delivery Network

Figure 1 outlines the overall architecture and high-level workflow of the proposed Function Delivery Network (FDN). The application developer provides an application *configuration specification* which describes the functions, APIs, permissions, configurations, and events, via a framework like Serverless. The *FDN Control Plane* then manages function deployment and data placement across target platforms, monitors the overall infrastructure and applications, and provides access control for authentication and authorization. We assume that the FDN has access to the container images (and kernel binaries for FPGAs) of application functions for various target platforms. Various behavioral models are constructed during functions invocations. These models are updated regularly in an online learning manner. The runtime decisions of function scheduling and data placement done by the *FDN Control Plane* is based on these models. The management of target platforms is done in a hierarchical manner, where the scheduling and placement decisions concerning the target platforms are taken by the *FDN Control Plane*, while the selection of the resources within the target platform is delegated to the *Sidecar Controller (SC)* within each target platform. Both the control plane and the local SC collaboratively work together to make the final decision. The functions invocations from the clients are

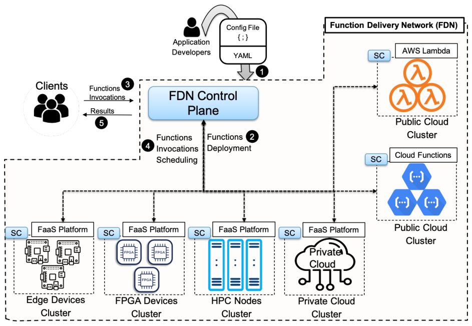


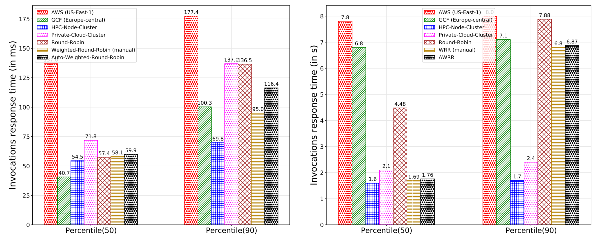
Fig. 1: Overall architecture and high level workflow of the Function Delivery Network (FDN).

load balanced by the *Courier* component of the *FDN Control Plane*, which automatically delivers the function invocations to the right target platform based on the various load balancing strategies. We designed a load balancing algorithm in addition to the naive Round Robin and manually set Weighted Round Robin (WRR) algorithms called Auto Weighted Round-Robin (AWRR). It uses *functions execution times* and *number of functions instances* metrics for each target across all FaaS functions for determining the weight of each target platform relative to the maximum execution time and the minimum number of function instances. The primary goal of the algorithms is to reduce the average response time across all requests.

III. Evaluation

Experimental Configuration. We evaluate the three function benchmarks : `primes-python` which calculates prime numbers till 10000000, `Sentiment-Analysis` which does the sentiment analysis of the given text, and `JSON-loads` which takes a large JSON file as input containing 3-D coordinates and returns the average coordinates value. The first two benchmarks are evaluated on and across four different target platforms: (i) Google Cloud Functions (GCF) and (ii) AWS Lambda for creating two *public cloud* platforms. (iii) The *private-cloud-cluster* is composed of three VMs hosted on a private cloud at the Leibniz Supercomputing Center (LRZ) [4], with each VM having four vCPU cores and 8 GiB of memory. (iv) *hpc-node-cluster* represents compute nodes from High Performance Computing (HPC) and is a dual-socket system, with each socket containing an Intel Cascade Lake processor with 22 cores. OpenWhisk on top of Kubernetes is used in both these private clusters. `JSON-loads` is evaluated for showcasing the energy efficiency on *hpc-node-cluster* and the *edge-cluster* consisting of three embedded Nvidia Jetson Nano devices which uses OpenFaaS on top of k3s.

Results. Fig. 2 shows the function invocation response times for the two microbenchmarks when (i) scheduled individually on target platforms, scheduled collaboratively across the different target platforms with (ii) naive round-robin, (iii) manually set WRR, and (iv) AWRR using the *Courier* load balancing. From the results, we can infer two conclusions: (i) a significant increase in performance (better SLOs) with



(a) `Sentiment-Analysis` (b) `primes-python`

Fig. 2: Comparison of function invocation response times for two microbenchmarks when they are scheduled on individual target platforms and collaboratively across target platforms using round-robin, WRR and AWRR load balancing.

TABLE I: Total energy consumption for *edge-cluster* and *hpc-node-cluster* target platforms when a workload is issued.

	Edge-Cluster			HPC-Cluster	
	N 1	**N 2**	**N 3**	**Socket0**	**Socket1**
Power WithoutLoad(W)	0.45	0.39	0.48	30.67	29.56
Power WithLoad(W)	1.41	2.04	0.95	37.39	37.01
CPU energy (J)		**2647.2**		**44645.64**	

collaborative scheduling across different target platforms as compared to when the functions are invoked exclusively on the individual platform with less resources and therefore collaboration in the FDN provides a method to overcome the shortcomings of individual target platforms. (ii) The developed AWRR algorithm can automatically determine the weights of each target platform based on different metrics and can provide a similar performance in comparison to manual WRR.

Table I shows the energy consumed by *edge-cluster* and *hpc-node-cluster* when a load of 400 requests per second is invoked on the function `JSON-loads` deployed on each one of them. Although, the P90 response time (6.32s) is higher for the *edge-cluster* as compared to *hpc-node-cluster* (2.3s), the total number of requests served is same for both target platforms. Therefore, if a client has a SLO P90 response time of seven seconds then both target platforms can be used for meeting it for this workload. However, choosing *edge-cluster* as the target platform for this workload saves a lot of energy.

Conclusion. Due to the current limitations of serverless computing for applications which are highly dynamic in their structure and computational requirements, we introduced the Function Delivery Network (FDN). We plan to extend it further to other heterogeneous computing devices.

References

[1] C. S. WG, "Cncf wg-serverless whitepaper v1. 0," https://gw.alipayobjects.com/os/basement_prod/24ec4498-71d4-4a60-b785-fa530456c65b.pdf, March 2018, [Online; Accessed: 15-July-2020].

[2] A. Jindal, M. Gerndt, M. Chadha, V. Podolskiy, and P. Chen, "Function delivery network: Extending serverless computing for heterogeneous platforms," *Software: Practice and Experience*, vol. n/a, no. n/a, 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2966

[3] "Aws iot greengrass - amazon web services," https://aws.amazon.com/greengrass/, (Accessed on 07/27/2020).

[4] "Lrz: Leibniz-rechenzentrum der bayerischen akademie der wissenschaften," https://www.lrz.de/, (Accessed on 07/30/2020).