MicroLambda – Packetized Computation for 5G Mobile Edge Computing

Saidur Rahman
Montana State University

Mike P. Wittie
Montana State University

Ahmed Elmokashfi
SimulaMet

Laura Stanley
Montana State University

Stacy Patterson
Rensselaer Polytechnic Institute

Immersive technologies enable Mixed Reality (MR), a form of spatial computing to blur the physical and digital creating a sense or presence. MR is well-positioned to be the next disruptive technology changing the way we work, live, think, and behave. MR over Mobile Edge Computing (MEC)/5G faces two unsolved challenges. The first is how to execute long computations on time-limited runtimes (for responsiveness) of serverless frameworks deployed on edge computing resources [1,2]. The second challenge is how to jointly perform 5G network scheduling and MEC computation placement.

We propose MicroLambda ($\mu\lambda$) – a method to partition computation across multiple serverless function invocations. $\mu\lambda$ partitions computation of arbitrary duration across multiple, sequential function (lambda) invocations that fit within the runtime timeout configured by MEC nodes. Thus, an offloaded MR process that might be too long for one lambda invocation can complete on $\mu\lambda$ using multiple *micro-lambdas* with the intermediate computation state passed between them. The advantage of $\mu\lambda$ is that it gives the 5G scheduler the flexibility to move the offloaded computation between MEC nodes. Effectively, the 5G scheduler may treat finite computation requests as packets, just as it does data transfer requests, and schedule them together to meet MR QoS requirements.

We assume an offloaded process p on an MEC node with user input i. p executes on a compute node with attached memory that stores intermediate process states $s_1, s_2, ..., s_{final}$ such as variables, or other data structures. After r_{actual} seconds of runtime p completes and returns output o. This abstraction does not map well to MEC computation, because r_{actual} might be greater than the runtime limit r_{limit} permitted on an MEC node. To complete an execution of p, $\mu\lambda$ partitions p's computation onto multiple lambda invocations. The first invocation of p accepts i and after fewer than r_{limit} milliseconds saves intermediate state s_1 onto an MEC storage node, for example a key-value store. The storage of intermediate state triggers subsequent invocation of p, which advances program state to s_2, and iteratively to s_{final} to return o. Each invocation of p and storage of intermediate state may take place at the same, or a different node as determined by a 5G scheduler.

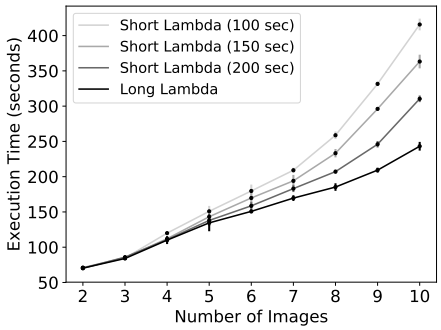

Figure 1: Execution time.

We compare the performance of an application running on the edge with and without $\mu\lambda$. The application supports a video monitoring scenario, where a camera (Client) captures video frames and transmits them to an edge compute device (Worker) to detect if a new frame contains a new, previously unseen face. This application requires Workers to save face encodings after detection and retrieve them at the start of the next lambda invocation from Redis. We implement this application using the Face Recognition library [3] based on the dlib machine learning toolkit [4]. The workload of our application is representative of a class of machine learning facial recognition applications being developed for MR [5]. Figure 1 shows the mean application execution time on $\mu\lambda$ and on a long-running lambda on the y-axis. The x-axis plots the size of the workload as the number of images. The r_{limit} of 100 s, 150 s, and 200 s allows a $\mu\lambda$ lambda to process from 2 to 5 images at a time depending on image size. The long-running lambda processes all the images within a given workload before returning. We base each data point on 40 trials and plot the corresponding 97% confidence intervals.

We presented $\mu\lambda$ – a flexible computation framework for offloading MR computation onto MEC. $\mu\lambda$ allows the splitting of stateful and long-running computation across multiple lambda invocations. The presented early results show comparative $\mu\lambda$ performance with respect to long-running lambdas. Future work will explore integration of $\mu\lambda$ within 5G network slicing and scheduling.

References

[1] Introducting serverless computing at the edge with Akamai EdgeWorkers. http://bit.ly/2P6JCxW, October 2019.

[2] EdgeWorkers User Guide: Limitations. http://bit.ly/2V46Bxs, February 2020.

[3] Adam Geitgey. Face recognition. https://github.com/ageitgey/face_recognition, December 2019.

[4] Dlib C++ library. http://dlib.net/, December 2019.

[5] Artsiom Ablavatski and Ivan Grishchenko. Real-Time AR Self-Expression with Machine Learning. http://bit.ly/385IpO7, March 2019.