The Promise and Challenges of Computation Deduplication and Reuse at the Network Edge

Md Washik Al Azad and Spyridon Mastorakis

ABSTRACT

In edge computing deployments, where devices may be in close proximity to each other, these devices may offload similar computational tasks (i.e., tasks with similar input data for the same edge computing service or for services of the same nature). This results in the execution of duplicate (redundant) computation, which may become a pressing issue for future edge computing environments, since such deployments are envisioned to consist of small-scale data centers at the edge. To tackle this issue, in this article, we highlight the importance of paradigms for the deduplication and reuse of computation at the network edge. Such paradigms have the potential to significantly reduce the completion times for offloaded tasks, accommodating more users, devices, and tasks with the same volume of deployed edge computing resources; however, they come with their own technical challenges. Finally, we present a multi-layer architecture to enable computation deduplication and reuse at the network edge, and discuss open challenges and future research directions.

INTRODUCTION

Edge computing has emerged as a paradigm to bring computing resources physically close to end users in an effort to address the increasing needs of applications for the low-latency processing of data generated by user devices, such as mobile phones, augmented reality (AR) headsets, and the Internet of Things (IoT) [1]. Edge computing deployments are envisioned to consist of small-scale data centers at the edge of the network [2]. At the same time, such deployments may target large-scale use cases (e.g., smart cities with hundreds of thousands or millions of residents). In such use cases, several devices may be in close proximity to each other, offloading tasks for “similar” computation (i.e., tasks with similar input data for the same edge computing service or services of the same nature) to the edge [3]. This can result in the execution of massive amounts of duplicate (redundant) computation, limiting the number of devices and tasks that can be accommodated by edge computing deployments. Overall, we expect the execution of duplicate and redundant computation to become a pressing issue for future edge computing deployments given the expected small scales of edge data centers and the need to accommodate large-scale use cases.

In this article, we highlight the promise of paradigms for the deduplication and reuse of computation at the network edge. In such paradigms, the results of previously executed tasks are stored with the goal to be reused and satisfy similar offloaded tasks, instead of executing similar tasks from scratch. The process of deduplication provides the means to infer whether reuse is possible by determining whether tasks similar to the offloaded ones have been previously executed and stored at the edge. As a result, such paradigms essentially “trade” storage for computing resources, having the potential to: significantly reduce the completion time of offloaded tasks; and accommodate more users, devices, and tasks with the same volumes of deployed edge computing resources. However, such paradigms come with their own technical challenges, which need to be addressed for their realization.

Our contribution in this article is two-fold:

• We highlight the promise that computation deduplication and reuse holds for edge computing environments and the need for paradigms for deduplication and reuse that consider the distributed nature of such environments, a key observation that prior research has overlooked.

• We present a multi-layer architecture to enable the pervasive computation deduplication and reuse at the network edge along with promising proof-of-concept evaluation results.

The rest of this article is organized as follows: in the next section, we discuss the importance of computation deduplication and reuse for edge computing deployments. Following that, we present the design of a multi-layer architecture for computation deduplication and reuse. Then we present open challenges and future research directions, and in the final section, we conclude our work.

Why Are Computation Deduplication and Reuse at the Edge Important?

With the projected growth of the number of IoT, mobile, and other devices at the edge, several devices may be in close proximity to each other. In such environments, redundant computation may occur, since temporal, spatial, and semantic correlation may exist between the input data of offloaded tasks. Devices may request the execution of the same services/processing functions.
offered at the edge with similar data as the inputs of these services/functions. In addition, available edge services may have processing components in common.

For example, a cognitive assistance application may be used on mobile phones or AR headsets to recognize the environment in the captured camera snapshots or AR scenes. In this context, visitors of famous sites all around the world may use this application to capture pictures/scenes of a site with their mobile phones or AR headsets. These pictures/scenes are offloaded to a nearby edge server where a cognitive assistance service (Fig. 1) processes the depicted site and returns information and content about the identified site to visitors (e.g., podcasts and videos related to the site, the story behind the site). In this scenario, visitors may request the same computation with similar inputs (e.g., pictures of the same site from different angles or distances), thus resulting in the same output (the received information and content about the same site). At the same time, this edge service may share processing components (e.g., object recognition, as illustrated in Fig. 1) with other edge services. Such services may include:

- A service used by an application that renders virtual furniture at certain positions to visualize furnished spaces [4]. Subsequently, an application for indoor navigation may render a virtual map to help users navigate buildings or stores. As a result, an edge service to process camera snapshots in this context may have a 3D graphics rendering component in common with the virtual furniture rendering service. Finally, in smart homes equipped with IoT devices, users can control these devices through voice commands. In such cases, residents of the same or nearby homes may invoke semantically similar commands that result in the same action (e.g., turning on the light in a room). To this end, the results of the corresponding edge service (Fig. 1) can be shared/reused among multiple users.

At the same time, given that next-generation applications may require ultra-low response times (e.g., AR may require response times less than 10 ms), the deduplication and reuse of computation can speed up the execution of tasks offloaded by user devices, since the execution results of previous similar tasks can be reused instead of executing computation-intensive tasks from scratch. This also ensures that the available edge computing resources are effectively utilized given their potentially limited scale by essentially trading storage for computing resources (i.e., to store and reuse previously executed tasks and their results). As a result, the reuse of computation has the potential to increase the capacity of edge computing deployments in

![FIGURE 1. Processing pipelines for four applications.](image-url)
FIGURE 2. A CCTV camera capturing snapshots of vehicle traffic, which are offloaded to the edge so that the number of cars in each snapshot is detected and the volume of traffic is estimated. Consecutive snapshots may be similar, thus yielding the same output once processed through a car detection service at edge servers and resulting in the execution of duplicate computation. To enable pervasive computation deduplication and reuse at the network edge, we propose a multi-layer architectural design.

GOALS AND TECHNICAL CHALLENGES
The fundamental goal to be achieved by solutions for pervasive reuse of computation at the edge is imposing minimal overheads on (potentially resource-constrained) user devices, the edge network infrastructure, and the edge servers so that users and edge computing providers can receive the substantial benefits of computation reuse. These benefits feature improved response times and the ability to accommodate increased numbers of tasks, users, and devices with a fixed amount of physical edge computing resources.

A Multi-Layer Architecture for Computation Deduplication and Reuse
In this section, we present a hierarchical architecture for the deduplication and reuse of computation at the network edge (Fig. 2). The first layer of the architecture consists of user devices, which can cache the results of previously offloaded tasks depending on the availability and capaci-
The primary goal of the edge network infrastructure is to forward tasks for the same service (or services with common components) and with similar input data to the edge server (among the available edge servers) that can maximize the chances of reusing previous computations. At the same time, in-network storage resources may be available to cache/store previously executed tasks and their execution results as they transit through the edge network infrastructure. When an offloaded task is received by a router, the router may exploit inter-frame similarity in Virtual Reality. The router may have the same hash value; such as structural similarity or cosine similarity.

Depending on the available resources, devices can cache offloaded tasks and the results of their execution (once received by edge servers). This can act as a first layer of computation reuse before new tasks are offloaded to the edge. For example, multiple applications may run on an AR headset, requiring the detection of objects in scenes captured by the headset (e.g., a driving assistance application that identifies potential accident conditions and informs drivers, and a smart navigation application that provides instructions to drivers on how to reach their final destinations) [4]. Such applications can essentially share and reuse the results of the tasks they offload. If deduplication and reuse are not possible at the device level (no similar previous tasks were found, or no resources are available on devices to store previously executed tasks and their results), a task along with its results may be reused only if the similarity between the input data of the task and the task that was reused is higher than this threshold. This offers flexibility and enables different applications to set different similarity thresholds that may be acceptable based on their requirements. Different forms of similarity can also be applied between different tasks as their similarity threshold may be selected by each application dependent on their requirements. Different forms of similarity can also be applied between different tasks as their similarity threshold may be selected by each application.

**Layer 1: User Devices**

The first layer of our architecture consists of user devices. Each device may run one or more applications and offload computational tasks to edge servers. Before offloading a task, each device needs to create a notion of how similar the task may be compared to previous tasks, essentially aiding all the layers of our architecture to find previous tasks that can be reused in a lightweight manner. This can happen through fast and space-efficient mechanisms, such as locality sensitive hashing (LSH) and feature hashing (FH). LSH is a technique that allows searching for the nearest neighbor of a data item by applying a hash function to its features. FH enables the vectorization of features extracted from data items by applying a hash function to the extracted features, while the resulting vector can be further hashed through LSH to cluster data items with similar feature values.

Through the application of such hashing techniques on the input data (e.g., images, videos, voice commands) of tasks: tasks with similar input data will likely be assigned the same hash value; fast similarity-based search operations will be enabled so that previous tasks with similar input data can be found and reused (no execution from scratch will be needed). In the example of Fig. 2, snapshots $n-1$ and $n$ may have the same hash values given and thus are highly similar. A minimum similarity threshold may be selected by each application so that a task can be reused only if the similarity between the input data of the task and the task that was reused is higher than this threshold. This offers flexibility and enables different applications to set different similarity thresholds that may be acceptable based on their requirements. Different forms of similarity can also be applied between different tasks as their similarity threshold may be selected by each application.

**Layer 2: Edge Network Infrastructure**

The primary goal of the edge network infrastructure is to forward tasks for the same service (or services with common components) and with similar input data to the edge server (among the available edge servers) that can maximize the chances of reusing previous computations. At the same time, in-network storage resources may be available to cache/store previously executed tasks and their execution results as they transit through the edge network infrastructure. When an offloaded task is received by a router, the router may store the results of the tasks they offload. If deduplication and reuse are not possible at the device level (no similar previous tasks were found, or no resources are available on devices to store previously executed tasks and their results), a task along with its results may be reused only if the similarity between the input data of the task and the task that was reused is higher than this threshold. This offers flexibility and enables different applications to set different similarity thresholds that may be acceptable based on their requirements. Different forms of similarity can also be applied between different tasks as their similarity threshold may be selected by each application.
TABLE 2: Task completion times when reuse occurs: at user devices and within the edge network infrastructure; and at edge servers.

| Dataset   | No reuse | Reuse (edge servers) | Reuse (devices and edge network) |
|-----------|----------|----------------------|----------------------------------|
| MNIST     | 120.42   | 21.23                | 8.82                             |
| Pandaset  | 116.65   | 19.52                | 7.26                             |
| Mobile AR | 106.64   | 13.32                | 6.68                             |
| CCTV      | 115.68   | 17.80                | 7.67                             |

A local storage resources are available. This similarity search process will take place based on the LSH or FH that has been attached by user devices to the offloaded task. If no previously executed task that can be reused is found, a router will forward a task based on its hash to an edge server. The space of the potential hash values is divided among the available edge servers so that each edge server is responsible for the execution of tasks with input data that falls under the range of the hash values assigned to this server. For example, in Fig. 2, each hash has a size of 4 B. To this end, the potential hash values will be between 0 and 65,535, while these values are equally divided among the available edge servers.

Layer 3: Edge Servers

Edge servers receive offloaded tasks and perform a nearest neighbor search to identify previously executed stored tasks that could be reused. Once the nearest neighbor of an incoming task t is found, an edge server will check whether the similarity between the input data of t and the nearest neighbor of t exceeds the minimum similarity threshold selected by the application that offloaded t. If this is the case, the found nearest neighbor task will be reused, and its results will be returned to the user in response to t. Otherwise, the server will execute t and store t and its execution results for potential reuse in the future. Each edge server maintains one or more hash tables, which index previously executed stored tasks based on the hashes of the tasks’ input data. Overall, the edge servers trade storage for computing to reduce the round-trip time (RTT) between devices and servers ranges between 12–16 ms. We have additionally deployed the task with a similarity threshold of 90 percent. Finally, in Fig. 3, we present the usage percentage of the CPU and memory resources of an edge server during the execution of 40,000 offloaded tasks. The results demonstrate that the usage percentage drops as the percentage of reuse increases. The resources of the edge server are also occupied for smaller amounts of time as the reuse percentage increases, since the execu-
controllers, which act as coordination points for managing the storage resources of user devices, edge routers, and servers. The hash value space needs to be divided and distributed among the available edge servers. In its current form, our proposed architecture could cause load imbalances among edge servers, since large amounts of similar tasks may be forwarded to the same edge server(s), thus increasing the load of certain servers and leaving others underutilized. In addition, mechanisms are needed to dynamically distribute the hash value space among edge servers. Our architecture could also benefit from techniques to optimize the usage of storage resources by storing tasks that are likely to be reused in the future while discarding ones that are not likely to be reused. Finally, the reuse of computation could be exploited by attackers to discover if tasks with similar data and/or for the same edge service have been previously executed. We further discuss these open issues and propose possible solutions in the following section.

**Open Challenges and Future Directions**

Computation deduplication and reuse show promise for edge computing environments, having the potential to improve response times. However, there are still open challenges to be addressed leading to several directions of future research.

**The “Curse” of Dimensionality**

Through hashing, high-dimensional data is converted to a fixed-size value. This process may require a large space of features for FH and a family of hash functions for LSH to be applied to the task input data in order to maintain satisfactory reuse accuracy. Large feature spaces and LSH function families may result in longer hashing and search times and increase the memory requirements for user devices, edge routers, and servers. Mechanisms such as hierarchical feature hashing [12] and multi-probe LSH [13] are required to keep the size of the feature space and the number of LSH functions manageable should be further explored.

**Distribution of Hash Value Space among Edge Servers**

The hash value space needs to be divided and distributed among the available edge servers. To achieve this, mechanisms of different nature (distributed and centralized) should be explored. Distributed mechanisms can enable servers to dynamically form a multicast communication group. In the context of this group, servers communicate directly to reach a consensus on how to divide the hash value space and which server will be responsible for which range of the hash value space. Locally centralized mechanisms may utilize software-defined networking (SDN) controllers, which act as coordination points for the distribution of the hash value space among servers. SDN controllers can inform edge routers about the distribution of the hash value space among servers, populating the reuse information table of routers. Initially, the hash value space can be equally distributed among servers and can be dynamically redistributed to balance the load among servers, as we describe below.

**Balancing the Load among Edge Servers**

As the space of potential hash values is divided among edge servers, load imbalances may occur. For example, if large amounts of tasks with similar input data are generated, all the tasks may be forwarded to the same server(s), increasing the load of certain servers while leaving other servers underutilized. This calls for mechanisms to achieve load balancing and reuse at the same time. For example, SDN controllers can monitor the computation reuse performance, the overhead among servers, and the load of servers and redistribute parts of the hash value space from one server to another to balance the load.

**Predicting the Likelihood of Reuse**

The storage resources of user devices, edge routers, and servers may have limited capacity. In environments that offer computation reuse, the different layers of the architecture may not be able to store the results of all executed tasks. To increase the impact and benefits of reuse, mechanisms to estimate/predict the chances of an executed task being reused in the future need to be explored. As a result, tasks not likely to be reused in the future may not be stored after execution, offering available storage space to tasks that are likely to be reused. Such mechanisms may also be essential in cases of tasks that consist of multiple sub-tasks (e.g., tasks that are formulated as a computation graph) to determine which sub-tasks to store and which ones to discard across the different layers of the architecture.

**Security and Privacy Implications**

Attacks can probe the edge architecture to discover if tasks for the same service and/or with similar input data have been previously executed. For example, attackers can offload tasks with images to be processed by an object detection edge service, while knowing that such tasks may need several tens or even hundreds of milliseconds to be executed. As a result, if the execution results are received much sooner, attackers can infer that a task with similar input data was reused. In addition, given that the execution results of tasks offloaded by different users can be shared/reused, solutions to isolate private results but
FIGURE 3 Percentage of CPU and memory usage of an edge server during the execution of 40,000 offloaded tasks (markers do not represent actual data points, but are used for better readability).

share non-private results in multi-tenant (multi-user) edge environments should be explored [14]. Attackers could also infer the locations of the devices that offload tasks [15]. The implications of reuse on the security and privacy of computations, the associated input data, and the location of devices should be further investigated.

SCALABILITY
Given the projected growth of devices and the wide spectrum of next-generation applications, scalability becomes a major challenge for computation deduplication and reuse architectures. Techniques to optimize the performance of hashing and nearest neighbor search operations can contribute to scaling up the number of tasks that can be handled. The scalability of computation reuse architectures can be further enhanced by performing reuse not only on the basis of individual applications, but for groups of applications that require the same type of data processing. For example, applications that need the detection of objects in images may invoke different services of the same type/nature deployed at the edge. Such services essentially provide the same type of data processing (i.e., detection of objects in images), however, they may achieve that through different object detection algorithms.

CONCLUSION
In this article, we present the promise and challenges of computation deduplication and reuse in edge computing deployments. We first present use cases that computation reuse can benefit, and we then discuss the technical challenges of realizing solutions for the reuse of computation. Moreover, we present the design of a multi-layer architecture for computation reuse and several open challenges and research directions. We believe that the effective management of the massive computation volumes projected to be produced at the edge will become a pressing issue; thus, reusing computation among devices, users, and applications will become a key mechanism to improve response times and accommodate additional users, devices, and tasks.

ACKNOWLEDGMENTS
This work is partially supported by NIH (NIGMS/P20GM109090), NSF under awards CNS-2016714 and CNS-2104700, the Nebraska University Collaboration Initiative, and the Nebraska Tobacco Settlement Biomedical Research Development Funds.

REFERENCES
[1] W. Shi et al., “Edge Computing: Vision and Challenges,” IEEE IoT J., vol. 3, no. 5, 2016, pp. 637–46.
[2] M. Satyanarayanan, “The Emergence of Edge Computing,” Computer, vol. 50, no. 1, 2017, pp. 30–39.
[3] P. Guo et al., “Fogycache: Cross-Device Approximate Computation Reuse,” Proc. 24th Annual Int’l. Conf. Mobile Computing and Networking, 2018, pp. 19–34.
[4] P. Guo and W. Hu, “Potluck: Cross-Application Approximate Deduplication for Computation-Intensive Mobile Applications,” Proc. 23rd Int’l. Conf. Architectural Support for Programming Languages and Operating Systems, 2018, pp. 271–84.
[5] U. Drolia et al., “Cacheir: Edge-Caching for Recognition Applications,” Proc. 2017 IEEE 37th Int’l. Conf. Distributed Computing Systems, 2017, pp. 276–86.
[6] J. Meng et al., “Coterie: Exploiting Frame Similarity to Enable High-quality Multiplayer VE on Commodity Mobile Devices,” Proc. 25th Int’l. Conf. Architectural Support for Programming Languages and Operating Systems, 2020, pp. 923–37.
[7] S. Matoriakos et al., “Cedges: When Edge Computing Meets Information-centric Networking,” IEEE IoT J., vol. 7, no. 5, 2020, pp. 4203–17.
[8] A. Andoni et al., “Practical and Optimal LSH for Angular Distance,” arXiv preprint arXiv:1509.02897, 2015.
[9] Y. LeCun, “The MNIST Database of Handwritten Digits,” 1998, http://yann.lecun.com/exdb/mnist/, accessed Mar. 10, 2021.
[10] “Pandaset by HESAI and Scale AI,” https://pandaset.org, accessed Mar. 10, 2021.
[11] M. Malar et al., “Interframe Coding of Canonical Patches for Low Bit-Rate Mobile Augmented Reality,” Int’l. J. Semantic Computing, vol. 7, 2013, pp. 5–24.
[12] B. Zhao et al., “Hierarchical Feature Hashing for Fast Dimensionality Reduction,” Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2014, pp. 2043–50.
[13] Q. Li et al., “Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search,” Proc. 33rd ACM Int’l. Conf. Very Large Data Bases, VLDB 2007, pp. 950–61.
[14] L. Chen et al., “Enclavecache: A Secure and Scalable Key-Value Cache in Multi-Tenant Clouds Using Intel SGX,” Proc. 26th Int’l. Middleware Conf., 2019, pp. 14–27.
[15] Z. Tian et al., “Location Privacy Challenges in Mobile Edge Computing: Classification and Exploration,” IEEE Network, vol. 34, no. 2, Mar./Apr. 2020, pp. 52–56.

BIOGRAPHIES
Md Washik Al Azad (malazad@unomaha.edu) is a Ph.D. student at the University of Nebraska Omaha. He received his B.S. in electronics and telecommunication engineering from the Rajshahi University of Engineering & Technology, Bangladesh in 2017. His interests include edge computing and security.
Spyridon Mastorakis (smastroakis@unomaha.edu) is an assistant professor in computer science at the University of Nebraska Omaha. He received his Ph.D. in computer science from the University of California, Los Angeles in 2019. His research interests include network systems and architectures, edge computing, and security.