Enabling Fine-Grained Edge Offloading for IoT

Vittorio Cozzolino
Technical University of Munich

Aaron Yi Ding
Technical University of Munich

Jörg Ott
Technical University of Munich

Dirk Kutscher
Huawei Technologies

ABSTRACT

In this paper we make the case for IoT edge offloading, which strives to exploit the resources on edge computing devices by offloading fine-grained computation tasks from the cloud closer to the users and data generators (i.e., IoT devices). The key motive is to enhance performance, security and privacy for IoT services. Our proposal bridges the gap between cloud computing and IoT by applying a *divide and conquer* approach over the multi-level (cloud, edge and IoT) information pipeline. To validate the design of IoT edge offloading, we developed a unikernel-based prototype and evaluated the system under various hardware and network conditions. Our experimentation has shown promising results and revealed the limitation of existing IoT hardware and virtualization platforms, shedding light on future research of edge computing and IoT.

1 MOTIVATION

Edge computing and Internet of Things (IoT) have become closely coupled in recent developments. IoT was initially conceived as extending the Internet with a new class of devices and use cases (e.g., constrained networks, personal devices). First architectures and frameworks introduced the notion of cloud-connected IoT deployments, with the assumption that most/all IoT edge networks need to be connected to the cloud, for example through some edge gateway and tunnel approach.

However, latest research and real application/deployment experience is challenging this slightly ossified model, because a strict dependency of cloud platform is not often desirable when: a) edge networks create data that needs to be accessed and processed locally, b) delay sensitivity does not allow for piping everything do the cloud and back, or c) the amount of data is too large to transfer to the cloud (in real-time) without causing congestion on backhaul.

With the diffusion of IoT networks and infrastructures, the risk of leaving behind and underestimating security threats is notable. This is a major concern when the information flows from IoT to the cloud is filled with highly sensitive information about end-users (e.g., Smart Health, Smart Grid). Therefore, adding additional security layers outside the cloud infrastructure could help enforce a tighter control over the IoT ecosystem. Performance and multi-tenancy supports would also benefit by adding intermediate units at the edge which possess local knowledge about the available physical resources. Thereby, we can optimize and maximize performance without requiring the cloud to oversee the entire process.

Effectively, IoT turns the traditional Internet traffic (data is primarily pushed downstream from the cloud to the consumer devices, by leveraging CDN etc.) around in a way that data gets produced at the edges, is consumed at the edges and (at least partially) sent to cloud-based platforms. This change of traffic model requires new

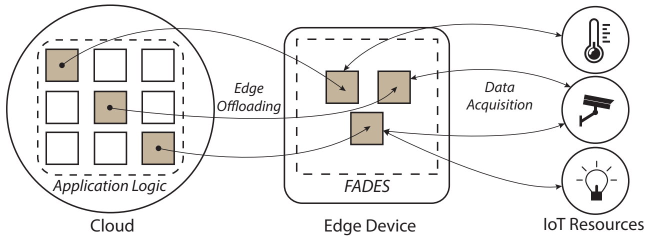


Figure 1: Edge Offloading

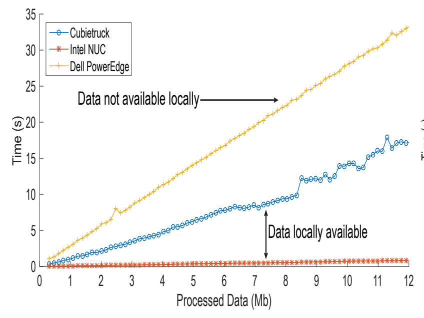
infrastructure support: edge computing could be the platform for that. In this regard, edge computation offloading can help build this multi-level (cloud, edge and IoT) information pipeline. Edge offloading, as shown in Figure 1, revisits the conventional cloud-based computation offloading where mobile devices resort to resourceful servers to handle heavy computation [2], to cater for the demands of IoT services. Therefore, our approach is reversed: we promote a paradigm where computation is sent by the server to constrained devices at the edge to take advantage of *data locality*. The key is to understand what to offload and this is strictly related to how the IoT is built.

Edge offloading is the perfect recipe to close the gap between cloud and IoT by applying a divide and conquer approach. Regardless of the back-end services, devices deployed at the edge of the network have to execute simple operation on data locally available. Therefore, by splitting a complex application into manifold simple and single-purpose tasks we can ship them in the shape of lightweight containers. Such approach is enabled by exploiting hardware resources via virtualization techniques.

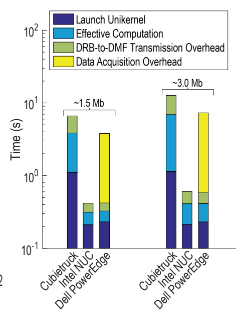
Virtualization and containerization technologies paved the way to efficiently exploit resources for multi-tenant environments. However, some of the existing technology is not applicable to IoT and dynamic scenarios. For example, ETSI MEC which is based on VMs and HTTP(S) traffic interception could be too heavy-weight and coarse-granular. What IoT needs is a more flexible, fine-grained, and modular platform. Therefore, we advocate the utilization of even lighter solutions like unikernels [1] [4] to help minimize as much as possible the transferred functionalities in order to, among other benefits, harden the system security and resiliency.

Our proposal is a modular system architecture designed to run compact, single purpose tasks on resource constrained edge devices. Instead of deploying full applications, our system aims at minimizing the offloaded code to edge devices in order to reduce the attack surface and optimize available resources utilization.

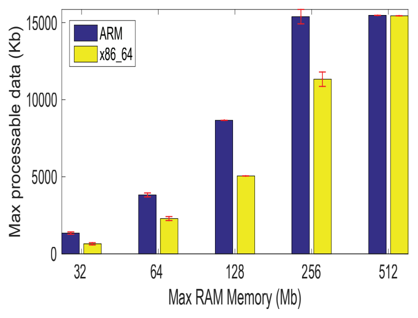
Given the existing work on computation offloading [2], our work differentiates itself by accurately tailoring the offloaded code/tasks



(a) Data Locality



(b) System Performance



(c) Data processing limits (memory)

in order to match the hardware constraints of the hosting device in IoT. Among other issues, we aim at solving the problem of multi-tenancy combined with resource utilization optimization.

2 TOWARD FINED-GRAINED EDGE OFFLOADING

As depicted in Figure 1, our design introduces an intermediate unit to enrich and augment the interaction between IoT resources and applications running in the cloud. The IoT resources offer physical capabilities to interact with the environment and carry out dedicated tasks. The back-end applications interact with our system by offloading parts of their operation logic. In our design, we consider the cloud as a repository of deployment-ready tasks designed for different IoT scenarios and purposes. For instance, the pollution control in smart cities is achieved by querying the pollution sensors, aggregating the information at the edge, and sending the final result to the cloud. In this regard, our system resembles a middlebox and oversees groups of IoT devices based on spatial proximity as Figure 1 indicates.

Our prototype is a unikernel-based system hosted by Xen hypervisor, which can be deployed across various IoT programmable boards. Our tool of choice is the MirageOS library operating system, which is specifically designed to build modular systems and runs natively on Xen [3]. The majority of functionalities and components are embedded into unikernels, and one orchestration module is developed in Python.

3 EXPERIMENTATION

The goal of our experimentation is to tackle the following questions: **Q1.** How much can edge deployed services benefit from data locality? **Q2.** How does our system perform under different workloads? **Q3.** How different architectures (x86, ARM) affect the performance of MirageOS? What are the unikernel PVM memory sizing requirements in relation to the amount of data to be manipulated?

For our tests, we selected three different devices: a Cubietruck, an Intel NUC and a Dell PowerEdge R520. We gleaned the data for the tests from our Intel Edison IoT testbed. The testbed is composed by 5 Intel Edison deployed in different office rooms on campus. Each board continuously collects environmental data through a set of sensors (humidity, temperature, light intensity, audio).

Figure 2a highlights how data locality can noticeably influence the system performance. Whenever the cloud application has to retrieve information from the edge, no matter how powerful it is, the amount of time required to fetch the data is much larger than offloading the computation to the edge device.

Figure 2b shows a detailed breakdown of the execution time for a task in our prototype. Four major factors add to the overall execution time. The factor of retrieving data from the edge only affects the scenario where the Dell PowerEdge server has to retrieve data from a remote network. The bars are grouped by amount of data to be processed. For example, the first group shows the performance for each device type given a payload of 1.5Mb.

The results of Figure 2a and 2b show that the presence of a sufficiently powerful device at the edge of the network combined with data locality makes edge offloading the best choice. Particularly, the Intel NUC outperforms the Dell PowerEdge while the Cubietruck is highly hindered by the overhead of intra-unikernel transmissions.

Figure 2c further shows the correlation between pre-allocated RAM and maximum amount of processable data. In our tests, we noticed that the device resources doesn't affect the maximum amount of processable data. Hence, the graph presents a generic comparison between ARM and x86.

4 DISCUSSION AND FUTURE WORK

The underlying idea of IoT edge offloading is to bridge the gap between complex applications running in the cloud and simple operations running at the edge. It's in this gap that we spot the opportunity to utilize lightweight virtualization e.g., unikernels, as an ideal vessel to ship single-purpose tasks for achieving modularity, flexibility and multi-tenancy. The experimentation over our prototype system has yielded useful insights for future research of IoT. Our next steps include: 1) evaluate our system scalability when running multiple offloaded task simultaneously, 2) extend and test our prototype with an industry-driven use case and 3) develop a programming model matching our system design.

REFERENCES

[1] Ketan Bhardwaj, Ming-Wei Shih, Pragya Agarwal, Ada Gavrilovska, Taesoo Kim, and Karsten Schwan. 2016. Fast, scalable and secure onloading of edge functions using AirBox. In *IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE.
[2] Kumar et al. 2013. A survey of computation offloading for mobile systems. *Mobile Networks and Applications* 18, 1 (2013), 129–140.
[3] Madhavapeddy et al. 2013. Unikernels: Library operating systems for the cloud. In *ACM SIGPLAN Notices*, Vol. 48. ACM.
[4] Madhavapeddy et al. 2015. Jitsu: Just-In-Time Summoning of Unikernels.. In *Proceedings of NSDI '15*.