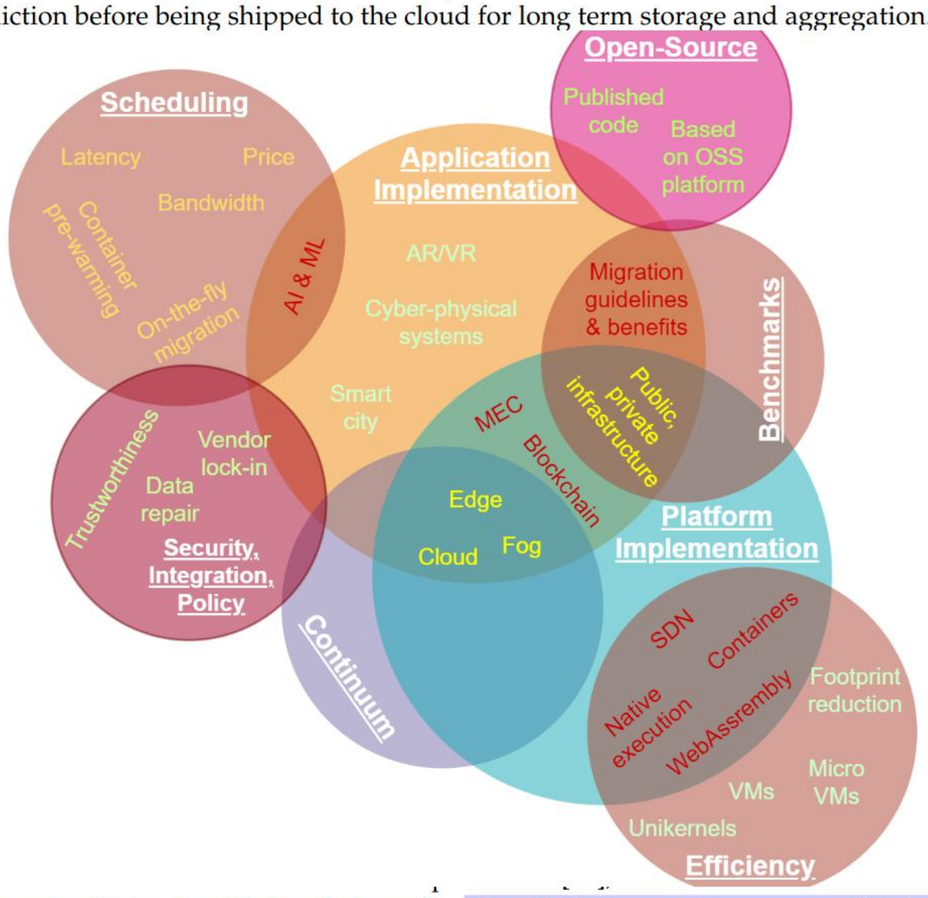
computing to the event driven IoT world. Utilizing serverless edge computing transforms the previously utilized ship-data-to-code paradigm, which incurred high network latency and transmission costs, to a ship-code-to-data paradigm [16]. Furthermore, by initially preprocessing the data at the edge, not only can network bandwidth be saved and faster response time obtained, but compliance with data protection laws can be ensured as well. In this manner, customer data can be anonymized closer to the data source, in the same jurisdiction before being shipped to the cloud for long term storage and aggregation.



higher adoption and outlining the benefits. Nonetheless, a recent survey on Stack Overflow, analyzing questions related to the topic of serverless computing [32], shows that the majority of encountered problems by developers are related particularly to application implementation. To solve this and to drive a higher level of adoption, formal guidelines should be published educating developers about the limitations of the network edge.

Efficiency improvements have been made to serverless edge platforms, trying to overcome the fact that existing serverless platforms developed initially for environments with plentiful resources are not a good fit for the resource constrained edge. The focus of this research area is finding alternative runtime environments that do not rely on containerization, thus avoiding the slow start-up incurred during the first invocation of a given function. A promising option is WebAssembly [52] with its portability and fast function start-up time [51], albeit further work is needed on improving the execution speed of the

deployed functions. Alternatives include the introduction of unikernels, a surprisingly under researched topic today, and the development of micro virtual machines [29], with some implementations already being open-sourced [88].

All these different workloads that have unpredictable load levels and need to cope efficiently with large increases in the number of requests emphasize the need for advanced resource allocation and scheduling algorithms that can better meet the FaaS quality of service (QoS) expectations during peaks [23]. A review of existing scheduling optimizations is offered in [27]. Even though it primarily focusses on the cloud, it is also relevant in network edge environments.

existing runtime architecture. Additionally, *AI & ML* is also present in both the Scheduling and Application Implementation categories. In the first case, AI is used in the process of workload scheduling, optimizing metrics such as latency, price or bandwidth and has no direct relation to the functionality of the instantiated applications whatsoever. However, in writing granular functions with a well defined role, and integrating their functionality to achieve more complex systems or applications. Thus, the main point of function as a service, and serverless computing in general, is to allow the developer to write functional code with a well defined task, using the desired programming language, which can then be uploaded and hosted as an atomic unit on a provider's infrastructure. This FaaS approach combined with common backend functionality such as databases, or message queues