# Accounting and Resource Scheduling at the Edge

Allaukik Abhishek
*Arm Research*

Chris Adeniyi–Jones
*Arm Research*

Eric Van Hensbergen
*Arm Research*

Marouane Balmakhtar
*Advanced Technologies and Innovation, T-Mobile*

## Abstract

The ownership model for Edge Compute is moving from single owner/single-tenant to multi-tenant/multi-owner. This trend follows the path taken in Cloud Computing where virtualization technology enables efficient sharing of physical hardware resources between cloud compute tenants. It is desirable for Edge Computing to be able to provide a similar sharing paradigm. However, in Edge computing deployments as well as the applications coming from different tenants, many of the other components i.e. Compute Nodes, Infrastructure, Connectivity and Networking, Edge Orchestration and management software, could be built and commissioned by legally separate operating entities. Sharing these multi-owner resources efficiently and securely is an active research topic but here we consider the questions around accounting - essential for determining who should be billed and what they are being billed for. In this paper, we look at various edge definitions and the resources that require metering and measuring. We try to draw parallels with the Cloud accounting model and highlight where it breaks and why. We then propose solutions which will help in resource allocation and accounting at the edge.

## 1 Introduction

End-users at the Edge may have significantly different Quality of Service (QoS) requirements. The QoS for a Cloud Gaming Solution which requires high bandwidth and intense compute is very different from an IoT sensor aggregation workload which requires scale, but only minimal bandwidth and compute requirements. Such distinctions entail that the resource requirements of the users are met in a deterministic fashion. This translates to (a) having a clear definition of the lowest minimum quantum for the resource (b) providing the quantum at the right time to meet the user's service level agreements (SLA) and (c) having a mechanism to independently verify the provided service levels for cross-verification.

The Edge Computing Landscape Q4 2019 update from Topio Networks, [Phi19], describes three Edge Computing trends: the building out of Edge Computing platforms is accelerating, many of these Edge platforms encompass a particular vertical and finally cloud-native application environments and deployment technologies are increasingly important for Edge computing. With the current trend of pushing more compute out to the Edge, the question of how to allocate resources and account for their use is becoming more urgent. This is compounded by the fact that the Edge is still the wild west in terms of hardware, software, peripherals, capabilities, and ownership. Consolidation is yet to happen. With such a wide spectrum, providing a common solution seems to be a difficult proposition. Yet, the accounting and billing terms need to be clearly articulated and tangibly measured to provide certainty (and to avoid future litigation!)

While there is still much uncertainty in how the Edge will look in the near term, there are unmistakable trends that could be useful to help define the resource allocation and scheduling issue. Once monolithic applications are being re-architected into smaller units of functionality - each unit now available to a unique scheduling policy. In this light, application orchestration frameworks like Kubernetes have emerged as leaders in workload deployment. Finally, data privacy and security play a key role in many design decisions, including peripherals, where bill-of-materials (BOM) costs now may no longer be the top priority.

## 2 Anatomy of the Edge

Figure 1 shows one of the many possible definitions of the multiple Edges that are relevant in Edge Computing: Device Edge, Access Edge, Aggregation Edge and Cloud Edge. The hardware on either side of these edge locations may be owned by different entities and the software (firmware, operating systems, middleware, applications) running at each of these locations may have been developed and deployed by another set of different entities.

Examples of this diverse ownership model are Telco provided Radio-Access-Network (RAN) Edge, Multi-Access Edge Compute (MEC) or Local-Breakout Edge.

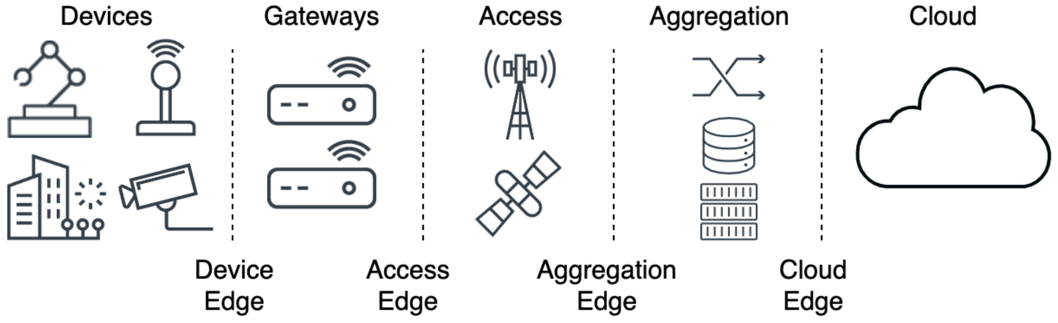

Figure 1: Generalized Architecture

Key workloads at the edge broadly fall in one of the these categories (a) Data normalization and compression (b) Message transformation and forwarding (c) Data processing pipelines and Storage. Various edge stacks have been designed to do one or many of these.

Each of these edge locations has characteristics that will have to be considered when attempting to deploy Edge compute for that location. These factors will have consequences for the scheduling and accounting.

At the *Device edge* the compute resources are likely to be very constrained. The amount of memory, number and capability of compute processing elements, amount of storage, and the amount of bandwidth available for edge computing will all be limited when compared to traditional shared computing environments. Added to this will be the possibility that this hardware may not be available 24/7 and that connectivity to the hardware may be unreliable. The device edge is also where we would expect to see a greater diversity in terms of hardware - for example an edge gateway may need to support several radio interface standards, each one requiring different device drivers and each one potentially providing a different software stack for applications to use. At the device edge the number of concurrently executing applications will be small and the software stack will be adapted to the constraints of limited resources.

At the *Access edge* the number of different radio/network interfaces required will be fewer than at the device edge. The hardware constraints for edge compute equipment on the access-side will also be less severe. Compute at this edge may be best served using converged (compute-in-network) hardware. An important requirement here is the ability to support the concurrent execution of applications from different customers whilst maintaining security and isolation and providing QoS for the overall network. At the access edge a single access point may be required to support up to 1000 concurrent edge compute applications.

At the *Aggregation edge* the hardware constraints will be less stringent than at the access edge. The major issues here will be to do with scale - managing the large flows of data and the orchestration of compute applications.

We would expect the *Cloud edge* hardware to have similar characteristics to Cloud/Datacenter instances and be capable of running a full cloud software stack.

A number of technologies can be used to provide execution environments for edge computing. These include sandboxed runtimes, container-based execution, and lightweight virtualisation solutions. The information required for scheduling and accounting will have to be generated by these execution environments.

## 3 Brief Review of Cloud Scheduling and Accounting

Resource allocation and subsequent accounting in the cloud are typically the jobs of a resource manager. The 3 key components that are required for efficient scheduling and accounting are resource discovery, work-load analysis, and the workload/resource scheduling unit.

The process starts with resource discovery. To create a resource list, different types of resources are identified and quantified. Hardware resources are divided into compute, network, and storage. The packaging of the resources could be homogeneous i.e. simple predefined VMs with a certain capability or heterogeneous i.e variety of virtualization technology, access to accelerators, mix of storage type. Other resources include things like software applications, operating systems, etc.

Workload analysis is the next step in the scheduling process. QoS, SLA, and availability define the general workload characteristics. QoS can be defined by guaranteed time for execution, max cost constraints, energy requirements, service guarantees, location, regulation, etc [VS19].

The scheduling unit takes input from discovered resource and workload characteristics and dispatches the workloads accordingly. The user-defined workload is broken up into a mixture of individual bite-size tasks and/or directed graphs of work units. The scheduler then does resource allocation, schedules tasks or work unit graphs, and executes [SC16].

The governor of the whole process is the monitoring unit which documents the execution time, monitors resource exhaustion and adds more if needed and finally cleans-up and releases resources. This then feeds into the accounting and billing unit which logs the details of the execution and charges the users.

In some cases , e.g , serverless workloads where the user wants to stay agnostic of details of underlying resources and is only concerned about the functionality, the accounting is decoupled from underlying resource provider [Wan+18]. The infrastructure is provided on functional demands and user time is accounted only during function execution.

## 4 Edge vs Cloud: Scheduling and Accounting

Edge, as contrasted in the previous section, diverges from the cloud in many respects. Therefore, the same accounting and scheduling techniques as applied in the cloud cannot be transplanted. Modifications in accounting and scheduling to match the edge infrastructure and edge use-cases is required. We discuss the key difference that mandates modifications in methodology.

a. Compute location: In cloud-based applications, the user is generally unconcerned by the location of the compute or the exactness of where the workload would execute. This is not the case with edge workloads. The user mandates the execute location. This severely limits the scheduler in the choice of resources that can be added for the user workload and adds strain to meeting QoS and SLA for the user.

b. Constrained Edge: Typically the edge is limited in resources both in terms of HW and SW capability. Guaranteeing QoS now not only depends on the user's workload, but also on the other tenants' functions. Factors like secure storage, energy usage, network bandwidth consumption are now nontrivial variables. With constraints, sand-boxing, as achieved in cloud, cannot be guaranteed.

c. Resource monitoring unit: The monitoring unit at the edge is also limited by the edge infrastructure. Accounting and scheduling can only be as good as the monitoring unit. Under constraints, the infrastructure does not possess adequate hooks to monitor the execution as present in the cloud. More over the monitor needs to be optimized to ensure the duty cycle balance between real workload execution and monitor tasks.

d. Multiple interfaces: In far edge scenarios, the interfaces attached increase in number and becomes much more varied. Managing ingress-egress from these new types of interfaces presents a different set of challenges. Resource allocation of these interfaces in a secure fashion and properly monitoring allocation constraints add to the accounting complexity at the edge.

e. Ownership and multi-tenancy: The ownership model for the edge is quite different from the cloud. The ownership splits between the HW infrastructure provider, network connectivity provider, real estate provider, and application developer. In the cloud, many of these are the same owner. Due to scale and resource growth constraints at the edge, the tenancy needs to be managed to avoid resource starving for low priority use-cases. The tenant bundling and preemption in itself becomes a complex algorithm.

## 5 Use-case: Operator Edge

In the Operator space, there is an emergence and adoption of concepts like control and user plane separation (CUPS) and local break-out networks [Tal+17]. Data handover to non-cellular technology like Wifi is leading to interesting Edge use-case scenarios. This is also being fueled by the advent of 5G technology which will, in principle, make Edge instantiation as simple as spinning up a VM.

The picture of Edge that is emerging as a very interesting ownership model. The connectivity and network layer ownership still resides with the operator. Hence it is the task of the operator to provide the basic data path from user-equipment (UE) to the edge. The bearer connection needs to be created based on UE identity, application types, QoS, etc. This can be statically negotiated with the UE and application provider or created dynamically based on rule-based filter matches.

At the Edge, the ingress and egress rules could be provided by the Operator or a 3rd party or Application Provider. The rules could be present in the form of Virtual Network Functions (VNFs) orchestrated and managed by the particular owner. As for the functions, these could be as simple as Forward / Reverse information base (FIBs), NAT rules, Firewalls.

An application that could be the sink and source of the UE data may be provided by another party. Moreover, the whole thing gets complicated if the Edge server itself is provided by a different entity either providing a bare-metal server or stack which supports some form of virtualization technique to allow applications and VNF orchestration. In terms of interfaces, this edge could possibly have other non-cellular pipe-outs like Ethernet or other wireless techniques like WiFi. The verbosity of ownership is huge. This implies accounting and billing at such an Edge is a complex problem. The accounting will encompass monitoring resource usage for each tenant - time slice, HW resources, VNFs, applications, and data in-n-out.

## 6 Use-case: Smart City Edge

The Smart City use-case is based around using Edge compute to provide methods of data-processing that are both privacy-preserving and efficient. A typical scenario would involve the deployment of an array of nodes with each node consisting of a variety of sensors such as environmental sensors, cameras, and microphones, along with control plane elements (for managing the node), compute elements (for processing the data) and communication elements (for sending results and control). An application deployed onto the node could, for example, use the sensor information to try and forecast if a location is likely to experience flooding. There is wide agreement that useful insights can be gained from data gathered at points within a city but there are also concerns in the details about how this data is processed: who has access to the data and what can they do with it? The edge nodes themselves may be owned by the city but the software stack may incorporate elements from different vendors. The applications themselves will also come from different sources. Additionally, Edge compute is useful here because it will not be feasible to send all the gathered data, such as high-definition video for example, from every node to a central cloud-based application.

In this use-case, there will be a number of different modes of scheduling for an application. Some applications will be event-driven - their execution triggered by changes in the sensor data. Some applications will be executed only periodically. Some applications will run continuously. There will also be variation in the number and types of sensors that each application requires. Some applications may require exclusive use of a sensor for a period. An edge platform may provide services to allow multiple applications to subscribe to data from a particular sensor. There will be variation in the compute intensity for each application. This will affect whether applications can be executed concurrently without exhausting the resources of the edge node itself. Applications may require some storage on the edge node - this too will need to be accounted for. Finally, applications will usually send information to a cloud-based service via connectivity provided by mobile (or in some cases wired) networking and this traffic will have to measure and accounted for.

## 7 Simple Edge Resource Allocation and Accounting Approaches

The content thus far has established that scheduling at the edge is *(a)* multi-dimensional problem *(b)* different from cloud data-centers. Also, Edge in 5G can be highly distributed which allows for various points of traffic monitoring and accounting that need to take place. To keep the problem bounded, for our work we made certain assumptions.

a. Type of workloads. For our initial experiments, we assume 2 class of workloads - run-to-completion and time-sliced. The distinction is made because the job quanta could be stateful or stateless. Initially, to simplify and not maintain internal states during preemption, we simplify it by allowing the job to run to completion. The jobs which are stateless can be preempted.

b. Resource quantization. This is mostly dependent on the current technology which may or may not allow the same resource sharing in a secure fashion. In that sense, the whole resource is allocated to the requesting job. The resource quotas policies are employed using constraints of the technology used.

c. Kubernetes as the orchestrator. We currently use Kubernetes as our orchestration engine and work with-in its constraints. All our assumptions on scheduling and monitoring are currently limited to either Kubernetes capabilities or the interface hooks provided by it [Aut20a]. For example, for scheduling, we look at working under the K8s framework but utilize the ability to recompile a custom scheduler. We employ the concepts of Pod Priority and Preemption but modify wherever relevant to match our study [Aut20b].

d. Monitoring and accounting. These pillars can be fulfilled assuming there is adequate visibility across the inbound and outbound streams of the edge. The edge paradigm which will use mostly virtual infrastructure may have different characteristics when it comes to interfaces which are used for virtualized monitoring and accounting functions. Traditional techniques of monitoring and accounting using standardized accounting and monitoring interfaces on physical nodes, hardware probes, and correlation with control plane and management plane information leave out the needs of the distributed nature of an edge environment due to much less overall visibility. This includes but not limited to the multiple data sources at various edge locations, the hidden data flows within or between virtual applications on the edge hosts, and flows within virtual switches and routers.

e. Accounting. Depending up on the business model, this may include any combination of accounting information of the applications and/ systems at the edge.

There is an increasing number of studies in grouping and scheduling techniques [Tsa+19] [Kan+18] [Raz+18]. Operating system process and thread schedulers, resource scheduling in network elements, wireless and multi-user spatial multiplexing. The matrix for edge is the maximum utilization of edge resources for maximum utilization and monetization. Hence it can be considered as a variant of MMSE or ZFS algorithms.

Working within these design assumptions and constraints, we propose two simple edge i) grouping and scheduling and ii) accounting mechanisms.

***Edge Grouping and scheduling***. An ideal match needs to be made between the target node and the tenant's application. As discussed the independent variables for grouping and scheduling could be location, latency, bandwidth and compute measurements. Gathering the appropriate resource category, the application requirements are matched with the resources in the class. In order to perform resource scheduling at the edge, we propose to use a simple weight matching to achieve it. The weight matching formula is below.

$$RS = (\sum ||x_r - y_r||W_i) \sum W_i \qquad (1)$$

where $x_r$ represents the attribute of application's requirement, $y_r$ represents the attributes of the available resources at the edge (edge capabilities/offerings/options), and represents the weights of the attributes. RS is resource scheduling of the app-edge matching score.

Different applications have different resource requirements. For different task preferences, they can be divided into location, latency, bandwidth, and compute requirements. As with resources, each task also consists of attributes, each of which occupies a different weight. In the above formula, the attribute that is required by the application and the resource attribute is calculated together, and the highest score that is obtained is returned to the application as the final result of the resource scheduling. The objective is to provide the mapping between application and node based on the application requirements and what the node capabilities are, such that the cost of the resource allocation is optimized. We propose an algorithm of the resource allocation problem at the edge. The core premise is to determine the best matching between application requirements and the edge capabilities by considering various parameter requirements such as location, compute, latency and bandwidth. The proposed algorithm has low complexity and adds negligible overhead to the execution of the resource allocation of applications at the edge. In subsequent additions to this position paper, we will attempt to evaluate the overall ‘fitness’ and ’fairness’ of the proposed algorithm in the context of resource allocation at the edge.

***Edge accounting***. Edge creates a new paradigm for the wireless network where network resources can be pushed to the edge logically for individual subscribers or applications. Monitoring traffic characteristics and performance (e.g., data rate, packet drop, latency), the end user’s geographical distribution, per session/subscriber/edge location instance-based monitoring, and more will be key to best use of the edge resources. Application performance performance metrics monitored and collected prior and during grouping and scheduling of the applications at the edge. Monitoring can help determine if the application performance metrics change such as latency and bandwidth, and will trigger notifications requesting for underlined edge resources to be updated (e.g., move application to another edge location, boost compute or memory resources at the current edge location, etc ...). Accounting and billing can use a combination of various criteria of edge resources, such as application usage (e.g., bandwidth), subscriber session usage (e.g., CDRs), to charge application providers’ usage systems. At the edge, each application will have specific resource requirements that it needs to run. Traffic accounting for each application needs to be computed to track for various aspects such as service quality assurance and billing purposes. For example, if the actual service quality does not meet the standard of the service level agreement, the appropriate party is automatically notified for further action. At a higher level, each application at the edge will be processing data traffic as part of the service offering. Hence, each application usage needs to be captured distinctly. The edge system may have more than one application running and processing traffic, hence it is important that the accounting takes into consideration application usage and not just the system usage. Depending up on the billing model requirements, the accounting should be available for each application as well as for the cumulative number of applications at the edge because there could be more than one application offered by the application provider to its subscribers. The application usage can be computed as a percentage of a resource dimension (e.g., bytes-in, bytes-out, session billing records, CPU usage). There are two sets of usage levels. The first usage level is application specific which can be used to bill a specific party using that application. The second usage level is a global which can account for system usage of a single or multiple edge systems. Each dimension has an associated weight that reflects the particular edge deployment. For instance, an edge deployment which has premium subscription characteristics such as high-security, high-reliability services (e.g., URLLC) will have higher weighted averages versus edge deployments that offer less demanding characteristics such as low-security, low-bandwidth services (e.g., mMTC)

$$ACS_k = \frac{(\alpha \sum app_a) \sum W_m + (\beta \sum app_g) \sum W_m}{m} \tag{2}$$

where $app_a$ represents the application specific accounting dimension, $app_g$ represents the global accounting dimension and $W_m$ represents the weights of the edge characteristic. $\alpha$ is an application factor and $\beta$ is a global factor. Depending up on the business accounting model, an application provider may not charge its subscribers the global (system) usage and will only charge the application consumption instead. Hence, $\beta = 0$ in this scenario. ACS is the Accounting Charging Score of the app-edge.

## 8 Discussion

The field of Edge resource scheduling, resource guarantee, monitoring and ultimately billing is ripe for research. Based on our analysis few topics that need immediate attention where the technical support is either non-existent or minimal:

**Interface monitoring and sharing** One of the most used resources at the edge will be external interfaces. This could be interfaced with communication resources like WiFi (IP based) or BLE, Zigbee, LoRa (non-IP). There is no standard method to guarantee and share resources. Protocols have some methods of QoS guarantees but translating it into Application-level SLA is missing. Also, there is a gap in methods that bridge the orchestration engine with such resources. Similarly, resources like the attached peripherals camera, sensors have no method for concurrent resource sharing other than at data level.

**Resource monitors** Several solutions exist for monitoring cloud-native *applications*. For example open-source tools such as Prometheus [Aut20c] and paid-for tools like Dynatrace [20]. Monitoring the underlying infrastructure is also possible with some of these tools. We note that the resources used by the monitoring solution are usually relatively small compared to the resources used by the applications. The resource constraints of compute at the Edge will require a different approach. It requires a much lighter overhead, support for heterogeneity and methods to cope with unreliability. In their paper "FMonE: A Flexible Monitoring Solution at the Edge" [Bra+18] the authors discussed the advantages of an edge-foccussed solution. Although there are some off-the-shelf mechanisms to monitor resources at the edge there is still room for further research. There is no single mechanism that allows the provider to capture all metrics reliably.

**Scheduling algorithms** We observed that there is a big gap in edge specific scheduling algorithms. The default methods are still very cloud-centric. As the Edge space matures there will be a need to have well-optimized methods for application grouping and scheduling to make the most of the edge resources.

## Bibliography

[SC16] Sukhpal Singh and Inderveer Chana. “A Survey on Resource Scheduling in Cloud Computing: Issues and Challenges”. In: *Journal of Grid Computing* (2016). URL: https://doi.org/10.1007/s10723-015-9359-2.

[Tal+17] T. Taleb et al. “On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration”. In: *IEEE Communications Surveys Tutorials* (2017).

[Bra+18] Álvaro Brandón et al. “FMonE: A Flexible Monitoring Solution at the Edge”. In: *Wireless Communications and Mobile Computing* 2018 (2018), p. 2068278. ISSN: 1530-8669. DOI: 10.1155/2018/2068278. URL: https://doi.org/10.1155/2018/2068278.

[Kan+18] T. Kan et al. “Task offloading and resource allocation in mobile-edge computing system”. In: *2018 27th Wireless and Optical Communication Conference (WOCC)* (2018).

[Raz+18] A. Razaque et al. “Distributed allocating algorithm based on cloud CPU scheme”. In: *2018 Fifth International Conference on Software Defined Systems (SDS)* (2018).

[Wan+18] Liang Wang et al. “Peeking Behind the Curtains of Serverless Platforms”. In: *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. Boston, MA: USENIX Association, July 2018, pp. 133–146. ISBN: ISBN 978-1-939133-01-4. URL: https://www.usenix.org/conference/atc18/presentation/wang-liang.

[Phi19] Gavin Whitechurch Philippe Cases. *Edge Computing Landscape Q4 2019 update*. Tech. rep. https://www.topionetworks.com/rep/edge-computing-market-2020-prospects-5dd722d678e002489009ac11. 2019.

[Tsa+19] M. Tsai et al. “Crucial-Resource Scheduling Strategy in Edge Computing”. In: *International Conference on Intelligent Computing and its Emerging Applications (ICEA)* (2019).

[VS19] Prateeksha Varshney and Yogesh Simmhan. “Characterizing Application Scheduling on Edge, Fog and Cloud Computing Resources”. In: *Wireless Communications and Mobile Computing* 2019 (2019). URL: https://doi.org/10.1002/spe.2699.

[Aut20a] Kubernetes Authors. *Kubernetes*. Feb. 2020. URL: https://kubernetes.io/docs/concepts/scheduling/kube-scheduler/.

[Aut20b] Kubernetes Authors. *Kubernetes*. Feb. 2020. URL: https://kubernetes.io/docs/concepts/policy/resource-quotas/.

[Aut20c] Prometheus Authors. *Prometheus is an open-source systems monitoring and alerting toolkit*. Feb. 2020. URL: https://prometheus.io.

[20] *Dynatrace is a software-intelligence monitoring platform*. Feb. 2020. URL: https://www.dynatrace.com/support/help/get-started/what-is-dynatrace/.