Edge Computing

Next Steps in Architecture, Design and Testing

Edge Computing: Next Steps in Architecture, Design and Testing

Introduction

While edge computing has rapidly gained popularity over the past few years, there are still countless debates about the definition of related terms and the right business models, architectures and technologies required to satisfy the seemingly endless number of emerging use cases of this novel way of deploying applications over distributed networks.

In our previous white paper the OSF Edge Computing Group defined cloud edge computing as resources and functionality delivered to the end users by extending the capabilities of traditional data centers out to the edge, either by connecting each individual edge node directly back to a central cloud or several regional data centers, or in some cases connected to each other in a mesh. From a bird's eye view, most of those edge solutions look loosely like interconnected spider webs of varying sizes and complexity.

In these types of infrastructures, there is no one well defined edge; most of these environments grow organically, with the possibility of different organizations owning the various components. For example, a public cloud provider might supply some of the core infrastructure, while other vendors are supplying the hardware, and yet a third set of integrators are building the software components. Trying to create a one size fits all solution is impossible for edge use cases due to the very different application needs in various industry segments. Interestingly, while cloud transformation started later in the telecom industry, operators have been pioneers in the evolution of cloud computing out to the edge. As owners of the network, telecom infrastructure is a key underlying element in edge architectures.

After four years, while there is no question that there is continuing interest in edge computing, there is little consensus on a standard edge definition, solution or architecture. That doesn't mean that edge is dead. Edge must be by its very nature highly adaptable. Adaptability is crucial to evolve existing software components to fit into new environments or give them elevated functionality. Edge computing is a technology evolution that is not restricted to any particular industry. As edge evolves, more industries find it relevant, which only brings fresh requirements or gives existing ones different contexts, attracting new parties to solve these challenges. Now more than ever, edge computing has the promise for a very bright future indeed!

This document highlights the OSF Edge Computing Group's work to more precisely define and test the validity of various edge reference architectures. To help with understanding the challenges, there are use cases from a variety of industry segments, demonstrating how the new paradigms for deploying and distributing cloud resources can use reference architecture models that satisfy these requirements.

Challenges in different industries

In a nutshell, edge computing moves more computational power and resources closer to end users by increasing the number of endpoints and locating them nearer to the consumers -- be they users or devices. Fundamentally, edge computing architectures are built on existing technologies and established paradigms for distributed systems, which means that there are many well understood components available to create the most effective architectures to build and deliver edge use cases.

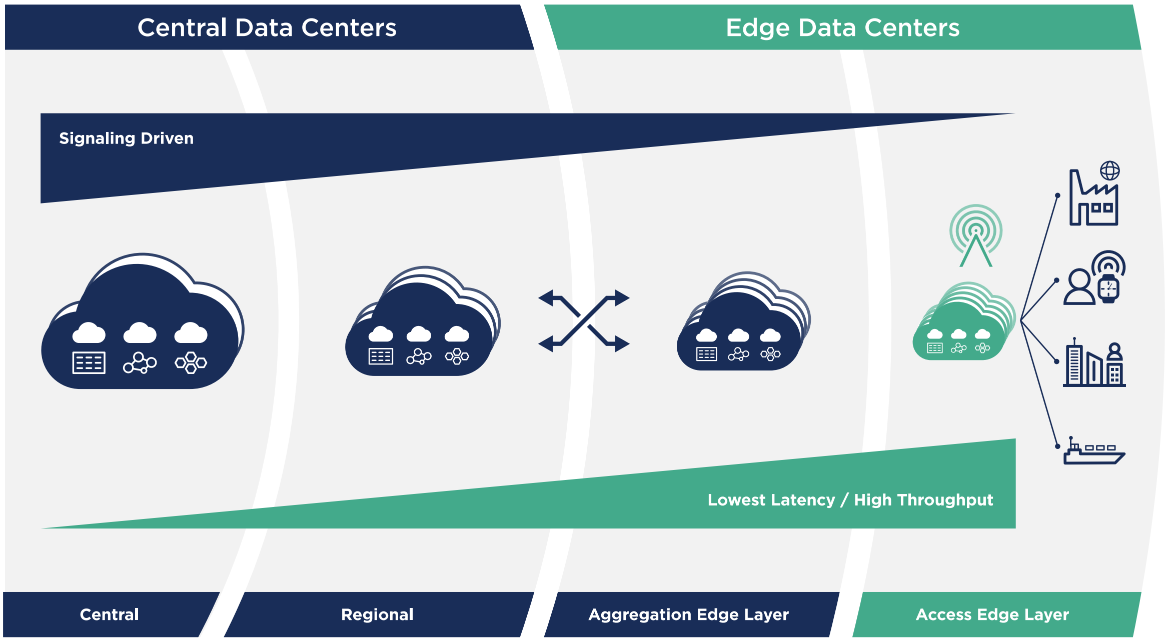
This section will guide you through some use cases to demonstrate how edge computing applies to different industries and highlight the benefits it delivers. We will also explore some of the differentiating requirements and ways to architect the systems so they do not require a radically new infrastructure just to comply with the requirements.

5G Brings You the Edge or Vice Versa?

5G telecom networks promise extreme mobile bandwidth, but to deliver, they require massive new and improved capabilities from the backbone infrastructures to manage the complexities, including critical traffic prioritization. The network needs to provide both high throughput and low latency combined with efficient use of the available capacity in order to support the performance demands of the emerging 5G offerings.

Signaling functions like the IMS control plane or Packet Core now rely on cloud architectures in large centralized data centers to increase flexibility and use hardware resources more efficiently. However, to get the same benefits for user plane and radio applications without bumping into the physical limitations of the speed of light, compute power needs to move further out to the edges of the network. This enables it to provide the extreme high bandwidth required between the radio equipment and the applications or to fulfill demands for low latency.

The most common approach is to choose a layered architecture with different levels from central to regional to aggregated edge, or further out to access edge layers. The exact number of levels will depend on the size of the operator network. The central locations are typically well equipped to handle high volumes of centralized signaling and are optimized for workloads which control the network itself. For more information about signaling workloads, reference Chapter 2.1 of the CNTT Reference Model under Control Plane for a list of examples. To increase end-to-end efficiency, it is important to pay attention to the separation of the signal processing and the user content transfer. The closer the end users are to the data and signal processing systems, the more optimized the workflow will be for handling low latency and high bandwidth traffic.



To describe what it all means in practice, take a Radio Access Network (RAN) as an example. Edge architectures require a re-think of the design of the Base Band Unit (BBU) component. This element is usually located near a radio tower site with computational and storage capabilities. In a 5G architecture targeting the edge cloud, a Cloud RAN (C-RAN) approach, the BBU can be disaggregated into a Central Unit (CU), a Distributed Unit (DU) and a Remote Radio Unit (RRU) where the DU functionality is often virtualized (vDU) with close proximity to the users, combined with hardware offloading solutions to be able to handle traffic more effectively. The illustration of the above edge architecture shows how the CU component can be located in an aggregated or regional edge site while the vDU would be located in the edge data centers. This setup allows more flexibility in managing the CU and DU while keeping the bandwidth utilization optimal, fulfilling the increasing user demands.

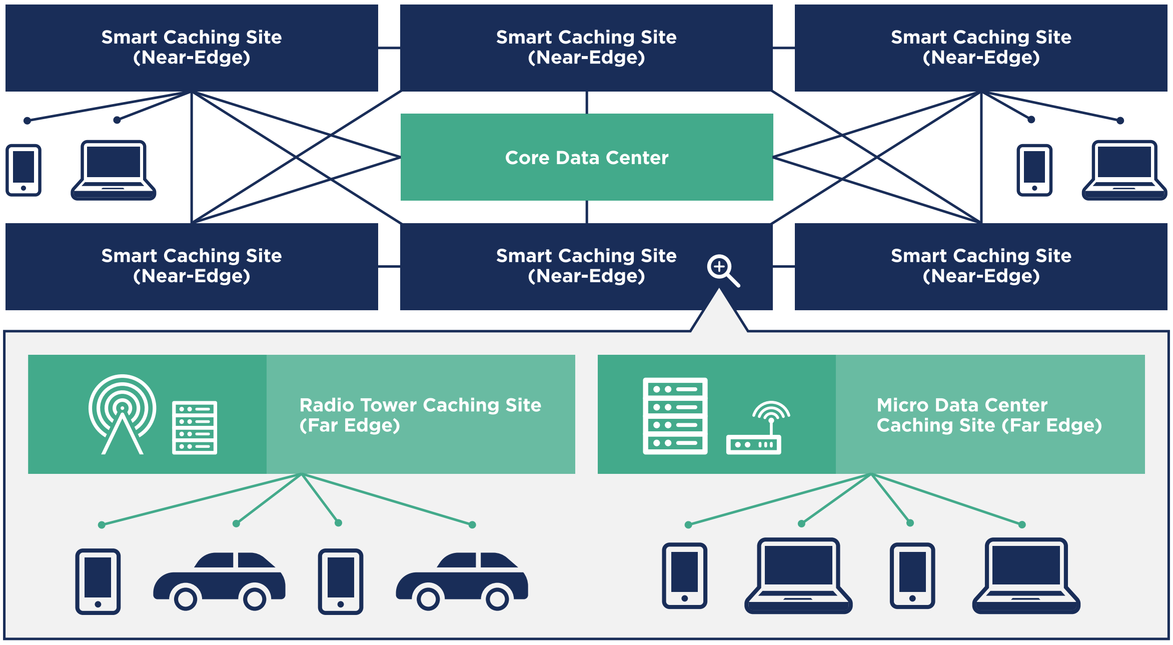
These architectural changes introduce new challenges for the lifecycle of the building blocks:

- **Automation**: to manage tens, hundreds or thousands of edge nodes
- **Remote provisioning**: allows the option to provision cloud infrastructure through WAN connection for sites at remote locations
- **‘Single pane of glass’**: a central dashboard to monitor the edge sites’ status including alarms and metrics
- **Remote upgrade**: edge sites are upgraded remotely with compatibility between the different versions of the software throughout the whole infrastructure
- **Resiliency**: the ability to run workloads without interruption in case of events like network connection disruption between data centers

Content Caching at the Edge

Reducing backhaul and latency metrics and improving quality of service (QoS) are good reasons for pushing content caching and management out to the network edge. A caching system can be as simple as a basic reverse-proxy or as complex as a whole software stack that not only caches content but provides additional functionality, such as video transcoding based on the user equipment (UE) device profile, location and available bandwidth.

Content delivery networks (CDN) are not a new concept. However, the creation of more CDN nodes with regional points of presence (PoP) are one of the first examples of what can now be considered near-edge-computing. With the explosion of video streaming, online gaming and social media, combined with the roll-out of 5G mobile networks, the need to push caching out to the far-edge has increased dramatically. The "last-mile" must become increasingly shorter to meet customer demand for better performance and user experience with these applications that are highly sensitive to network latency. This is encouraging content providers to migrate from a traditional, regional PoP CDN model to edge-based intelligent and transparent caching architectures.


Smart Caching Site (Near-Edge)
Smart Caching Site (Near-Edge)
Smart Caching Site (Near-Edge)
Core Data Center
Smart Caching Site (Near-Edge)
Smart Caching Site (Near-Edge)
Smart Caching Site (Near-Edge)
Radio Tower Caching Site (Far Edge)
Micro Data Center Caching Site (Far Edge)

The Pareto Principle, or 80-20 rule, applies to video streaming; that is, 80% of customers will only consume 20% of the available content. Therefore, by only caching 20% of their content, service providers will have 80% of traffic being pulled from edge data centers. This greatly reduces load on backbone networks while improving user experience.

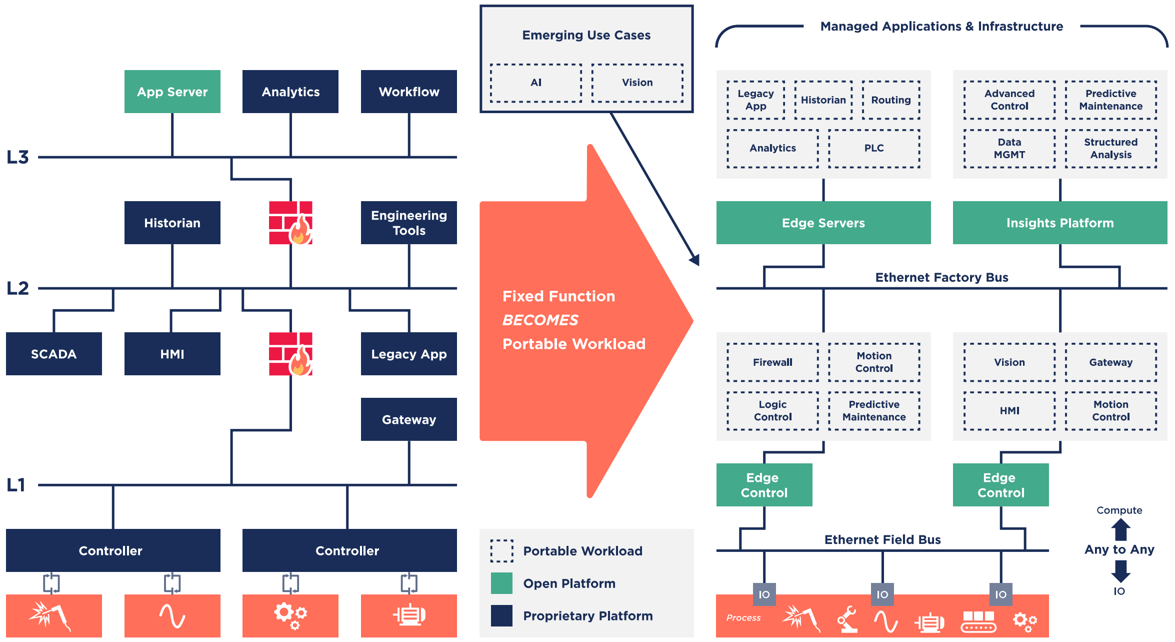
Caching systems in edge environments need to take end user device (EUD) proximity, system load and additional metrics as factors in determining which edge data center will deliver the payloads to which endpoints. In recent prototypes, smart caching frameworks use an agent in the central cloud that redirects content requests to the optimum edge data center using algorithms based on metrics such as UE location and load on the given edge site.

Manufacturing in the Digital Era

Industry 4.0 is often identified with the fourth industrial revolution. The concept is that factories are using computers and automation in new ways by incorporating autonomous systems and machine learning to make smarter factories. This paradigm shift includes the use of open hardware and software components in the solutions.

Factories are using more automation and leveraging cloud technologies for flexibility, reliability and robustness, which also allows for the possibility of introducing new methods such as machine vision and learning to increase production efficiency. The amount of data processing and computational power needed to support these technologies is increasing by orders of magnitude. Many applications move the data from the factory floor to a public or private cloud, but in many cases the latency impacts and transmission costs can lead to disruptions on the assembly line. To fulfill the high performance and low latency communication needs, at least some of the data processing and filtering needs to stay within the factory network, while still being able to use the cloud resources more effectively. Further processing of the data collected by various sensors is done in the centralized cloud data center. Reusable portable microservices located at the edge nodes fulfill tasks that are part of new vision applications or deep learning mechanisms.

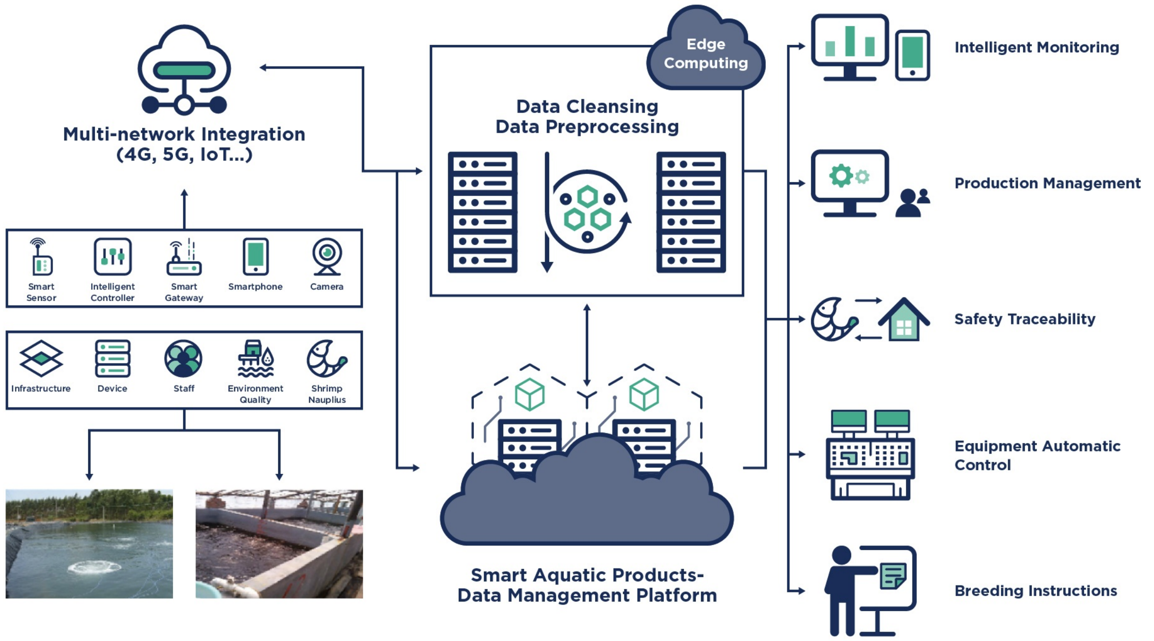
Similarly to the telecommunication industry, manufacturing also has very strict requirements. To fulfill the control systems’ real-time and functional safety needs, they can use technologies such as Time Sensitive Networking (TSN) on the lower layers of the architecture.



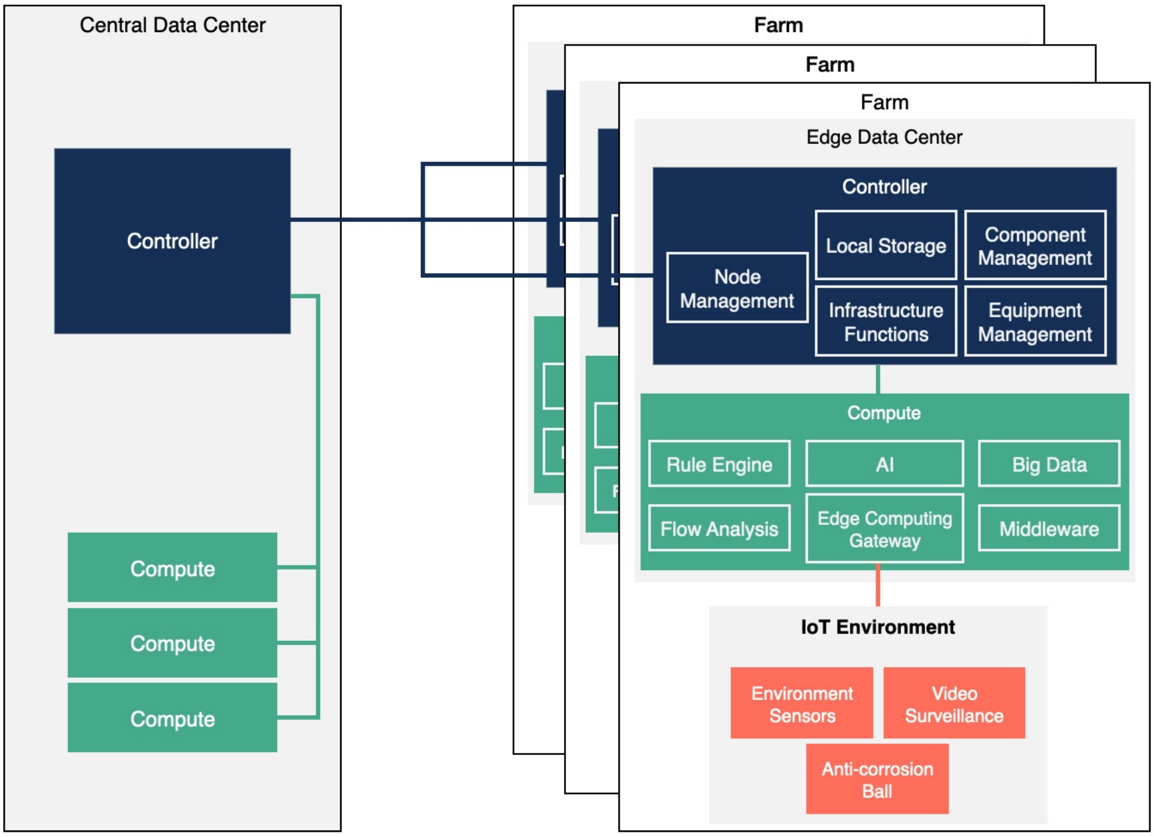
Edge Computing for Intelligent Aquaculture

Aquaculture is similar to agriculture, except that instead of domestic animals, it breeds and harvests fish, shellfish, algae and other organisms that live in a variety of salt or freshwater environments. These environments can be very fragile; therefore, it requires high precision to create and sustain healthy and balanced ecosystems. To increase production while providing a safe and healthy environment for the animals, automation is highly desirable. This use case is also a great example of where equipment is deployed and running in poor environmental conditions.

This section describes shrimp farms, which are controlled ecosystems where humans and automated tools oversee the entire lifecycle of the animals from the larva phase to the fully grown harvestable stage. The systems even follow the transportation of the shrimp after they are harvested. Like agriculture, the environmental conditions highly affect the animals’ conditions, and therefore the ponds need to be closely monitored for any changes that might affect the well-being of the shrimp, so that prompt actions can be taken to avoid loss.



The architecture diagram below shows a detailed view of the edge data center with an automated system used to operate a shrimp farm.



Some of the system functions and elements that need to be taken into consideration include:

- **Environmental monitoring**, including data collection and reporting of metrics such as: outdoor temperature and humidity, water quality, PH value and temperature, dissolved oxygen, ammonia, nitrogen, and nitrite
- **Video surveillance** for illegal intrusion detection and compliance identification of staff using clothing and face recognition
- **Smart breeding** that includes automated feeding and inventory tracking (food, medicine, auxiliary materials, disinfectant and so forth)
- **Platform traceability** to query and display the entire supply chain to ensure high quality of aquatic products

By automating and connecting these farms, the solution minimizes the isolation that exists in this industry. The platform provides data to be collected and analyzed both locally on the farms and centrally to improve the environmental conditions and prevent mistakes while using chemicals like auxiliary materials and disinfectants.

With more computational power at the edge data centers, it is possible to store and analyze local monitoring data for faster reaction time to manage changes in environmental conditions or modify feeding strategy. The system can also pre-filter data before sending it to the central cloud for further processing. For instance, the system can pre-process water quality data from the monitoring sensors and send structured information back to the central cloud. The local node can provide much faster feedback compared to performing all operations in the central cloud and sending instructions back to the edge data centers.

Digitalization has already provided much innovation, but there is still room for improvement, such as reducing the labor costs related to collecting data and improving data analysis to be faster and more reliable. With edge computing techniques, it is possible to build intelligent aquaculture infrastructure in order to introduce artificial intelligence and machine learning techniques that will optimize feeding strategy or reduce cost by minimizing human error and reacting faster to machine failures.

Technology Considerations

As can be seen from these few use cases, there are both common challenges and functionality that become even more crucial in edge and hybrid environments. As use cases evolve into more production deployments, the common characteristics and challenges originally documented in the “Cloud Edge Computing: Beyond the Data Center” white paper remain relevant.

The highest focus is still on reducing latency and mitigating bandwidth limitations. Further similarity between the different use cases, regardless of the industry they are in, is the increased demand for functions like machine learning and video transcoding on the edge. Due to the throughput demands of applications like these and workloads such as virtual network functions (VNF) for 5G, various offloading and acceleration technologies are being leveraged to boost performance through software and hardware, such as:

- Single-root input/output virtualization (SR-IOV): This technology allows VMs and containers to share direct access to a distinct part of a device, such as network adapters, using the PCI Express interface.
- Data Plane Development Kit (DPDK): DPDK allows higher network packet throughput by using offloading and

scheduling techniques. While this is not a technology specific for edge it is crucial to enable the option to use it to fulfill strict requirements in highly resource constrained environments.

- Non-uniform memory access (NUMA): This is another method to increase throughput by allocating dedicated memory blocks to an instance. For further optimization the memory block is local to the CPU core on which the instance is working. For very small edge sites this could become a bottleneck, depending on the edge site workload mix.
- SmartNics/Field-programmable gate array (FPGA): It is a hardware acceleration option that is already used for vRAN deployments to increase the performance of sites with compute-intensive workloads. The FPGA units are programmed with workload-specific software to offload the execution of some application-specific algorithms.
- Graphics Processing Unit (GPU): GPUs have a high-number of cores which may be utilized by a wide variety of parallel-processing intensive workloads such as MapReduce, machine learning (ML), IoT and gaming. While using GPUs is a good way to increase the system performance where the workload demands it, it introduces the question of cost constraints, especially if the number of edge sites starts to grow.

Reference Architectures

Architecture design is always specific to the use case, taking into account all the needs of the given planned workload and fine tuning the infrastructure on demand. As discussed earlier, there is no single solution that would fulfill every need. However, there are common models that describe high-level layouts which become important for day-2 operations and the overall behavior of the systems.

Before going into detail about the individual site type configurations, there is a decision that needs to be made on where to locate the different infrastructure services' control functions and how they need to behave. These models and decisions are not specific to the technologies nor do they depend on the particular software solution chosen.

The use cases in this document are mostly envisioned as a spider web type of architecture with hierarchy automatically able to scale the number of endpoints. Depending on needs, there are choices on the level of autonomy at each layer of the architecture to support, manage and scale the massively distributed systems. The network connectivity between the edge nodes requires a focus on availability and reliability, as opposed to bandwidth and latency.

This section covers two common high-level architecture models that show the two different approaches. They are the Centralized Control Plane and the Distributed Control Plane models. Since this is a high-level discussion, the assumption is that there will be enough compute, storage and networking functionality to the edge to cover the basic needs; any specialized configurations or features are out of scope. The architecture models also show required functionality for each site but do not discuss how to realize it with any specific solution such as Kubernetes, OpenStack, and so forth. However, aspects and tools that were considered during the development of the models include:

- Challenges of managing a large number of edge data centers: Available functionality at the edge data center vs. orchestration overhead
- Preparing the architecture to handle one failure at a time: e.g.: Network connection loss or degradation to the central or regional data center
- Providing minimal viable functionality on small footprints

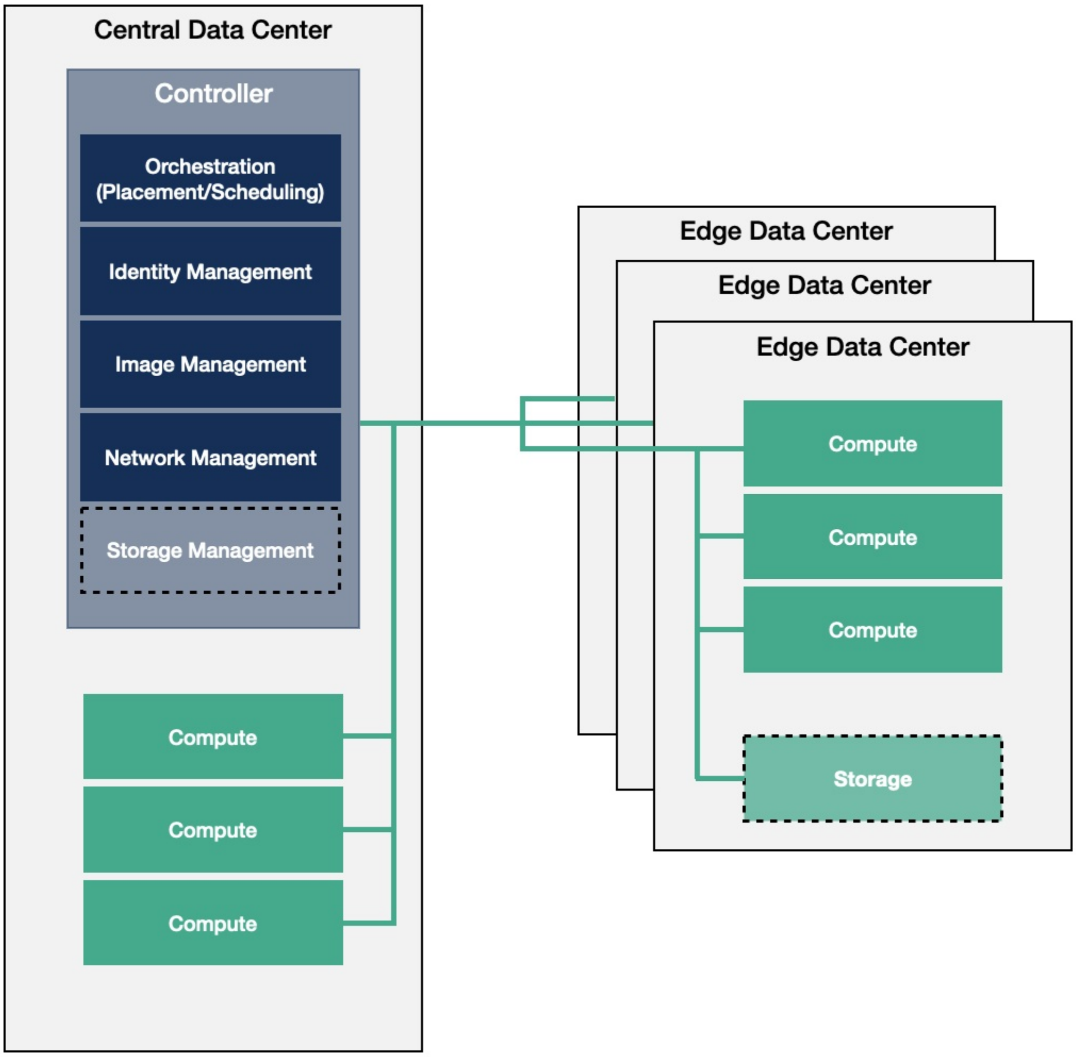
There are other studies that cover similar architectural considerations and hold similar characteristics without being fully aligned with one model or the other. For instance, a recent study presents a disruptive approach consisting of running standalone OpenStack installations in different geographical locations with collaboration between them on demand. The approach delivers the illusion of a single connected system without requiring intrusive changes.

Discussing and developing additional details around the requirements and solutions in integrating storage solutions and further new components into edge architectures is part of the future work of the OSF Edge Computing Group.

Centralized Control Plane

For the Centralized Control Plane model, the edge infrastructure is built as a traditional single data center environment which is geographically distributed with WAN connections between the controller and compute nodes. If a distributed node becomes disconnected from the other nodes, there is a risk that the separated node might become non-functional.

Due to the constraints of this model, the nodes rely heavily on the centralized data center to carry the burden of management and orchestration of the edge compute, storage and networking services because they run all the controller functions. Compute services incorporate running bare metal, containerized and virtualized workloads alike. Related functions which are needed to execute the workload of the infrastructure are distributed between the central and the edge data centers.



The diagram above shows that all of the key control functionality is located in the central site, including all identity management and orchestration functions. If you set aside the geographically distributed nature, this approach faces very similar challenges as operating large-scale data centers. On the plus side, it provides a centralized view of the infrastructure as a whole, which has its advantages from an operational perspective.

While the management and orchestration services are centralized, this architecture is less resilient to failures from network connection loss. The edge data center doesn't have full autonomy, therefore distributing configuration changes might fail if there is lost access to the image library or the identity management service. The configuration needs to allow applications to continue running even in case of network outages if the use case requires the workload to be highly available, i.e. a Point of Sales system in a retail deployment or the industrial robots operating in an IoT scenario. This can be challenging because most data center centric deployments treat compute nodes as failed resources when they become unreachable. In addition the Identity Provider (IdP) service can either be placed in the central data center or remotely with connection to the identity management service which limits user management and authentication. Depending on the situation, this might be considered more secure due to the centralized controllers, or less flexible because it might mean lost access by users at a critical juncture.

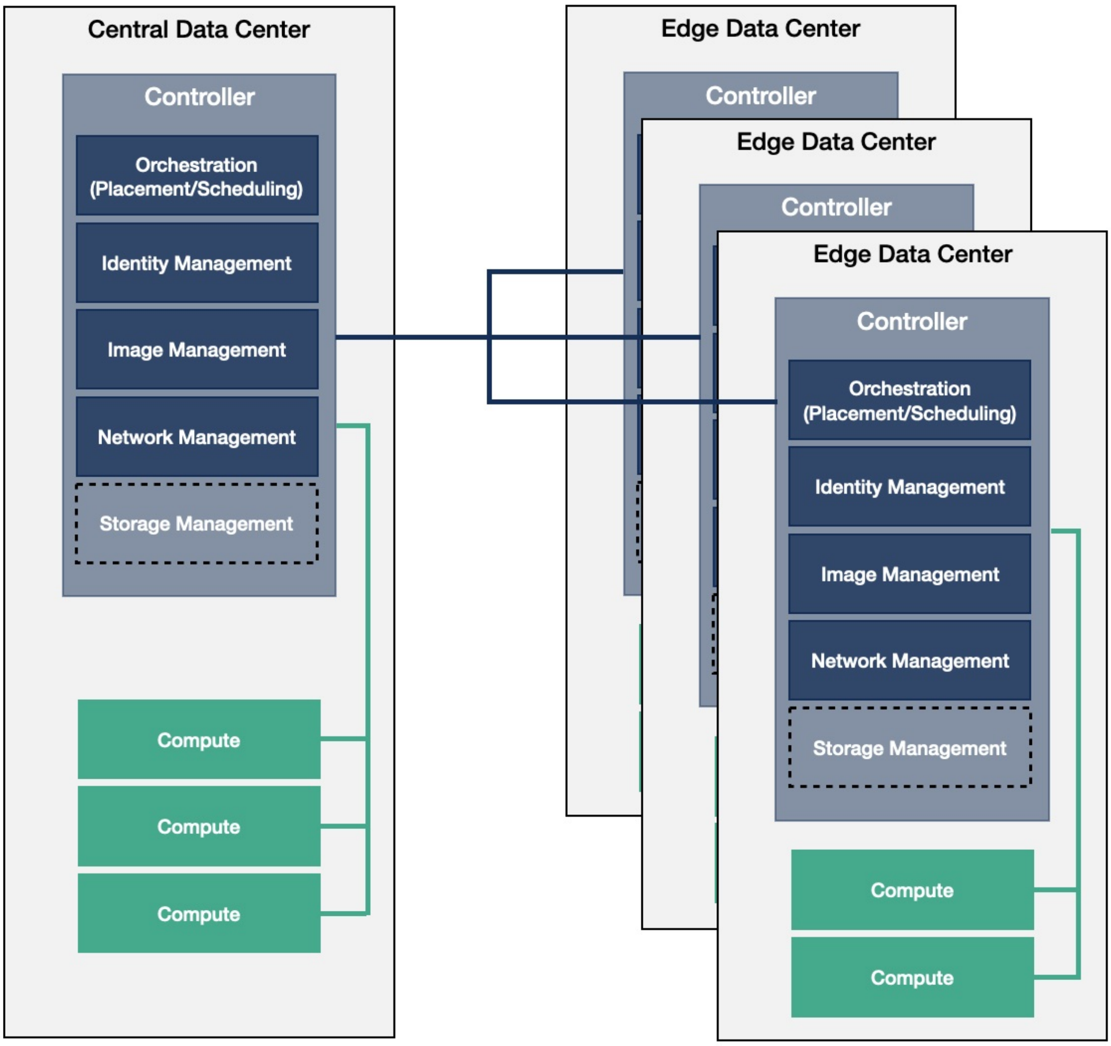
Typically, building such architectures uses existing software components as building blocks from well-known projects such as OpenStack and Kubernetes. Some edge sites might only have containerized workloads while other sites might be running VMs. It is recommended to review the Distributed Compute Node (DCN) deployment configuration of TripleO which is aligned with this model.

In summary, this architecture model does not fulfill every use case, but it provides an evolution path to already existing architectures. Plus, it also suits the needs of scenarios where autonomous behavior is not a requirement.

Distributed Control Plane

A larger set of use cases demands edge sites to be more fully functional on their own. This means they are more resilient to network connectivity issues as well as being able to minimize disruption caused by latency between edge sites.

The Distributed Control Plane model defines an architecture where the majority of the control services reside on the large/medium edge data centers. This provides an orchestrational overhead to synchronize between these data centers and manage them individually and as part of a larger, connected environment at the same time.



There are different options that can be used to overcome the operational challenges of this model. One method is to use federation techniques to connect the databases to operate the infrastructure as a whole; another option is to synchronize the databases across sites to make sure they have the same working set of configurations across the deployment. This model still allows for the existence of small edge data centers with small footprints where there would be a limited amount of compute services, and the preference would be to devote the majority of the available resources to the workloads.

The most common example is when the location of the components of the identity management service are chosen based on the scenario along with one of the aforementioned methods to connect them. The choice depends on the characteristics of the individual use case and the capabilities of the software components used, because the overall behavior and management of each configuration is different. For instance, using the OpenStack Identity Management service (Keystone) to locate it into an edge deployment without the limitation of technologies as its API supports both OpenStack and Kubernetes or the combination of both.

This architecture model is much more flexible in case of a network connection loss because all the required services to modify the workloads or perform user management operations are available locally. There are still potential obstacles, such as not having all the images available locally due to limitations of storage and cache sizes. There are also new challenges due to the additional burden of running a large number of control functions across a geographically distributed environment that makes managing the orchestration type services more complex.

As in the previous case, this architecture supports a combination of OpenStack and Kubernetes services that can be distributed in the environment to fulfill all the required functionality for each site. An example of this is StarlingX, as its architecture closely resembles the distributed model.

There are hybrid solutions on the market that try to leverage the best of both worlds by deploying full installations in the central nodes as well as large/medium edge data centers and have an orchestration type service on top, such as ONAP, an orchestration tool used in the telecom industry.

Future architectural considerations

The above described models are still under development as more needs and requirements are gathered in specific areas, such as:

- **Storage**: Considerations include local storage to enable high performance and low latency processing of data as well as providing options to connect to remote storage solutions. Several alternatives are available, ranging from small and simple systems like software RAID or LVM to large and highly reliable distributed storage managers like

Ceph.

- **Bare metal management**: It can be introduced on multiple layers, one being the infrastructure operator in need of managing and scaling their infrastructure to include zero touch provisioning methods, and the other being the user of the infrastructure who may get the permission and option to create new edge sites on demand.

There are other studies that cover similar architectural considerations and hold similar characteristics without being fully aligned with one model or the other. For instance, a recent study presents a disruptive approach consisting of running standalone OpenStack installations in different geographical locations with collaboration between them on demand. The approach delivers the illusion of a single connected system without requiring intrusive changes.

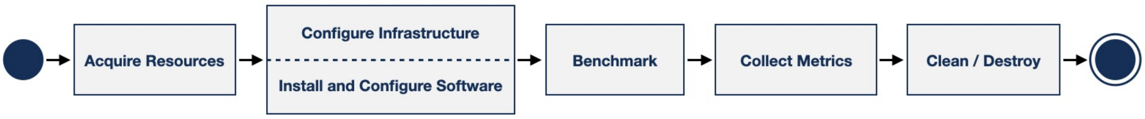
Discussing and developing additional details around the requirements and solutions in integrating storage solutions and further new components into edge architectures is part of the future work of the OSF Edge Computing Group.

Testing considerations

Defining common architectures for edge solutions is a complicated challenge in itself, but it is only the beginning of the journey. The next step is to be able to deploy and test the solution to verify and validate its functionality and ensure it performs as expected. As the edge architectures are still in the early phase, it is important to be able to identify advantages and disadvantages of the characteristics for each model to determine the best fit for a given use case.

The building blocks are already available to create edge deployments for OpenStack and Kubernetes. These are both open source projects with extensive testing efforts that are available in an open environment. While it is common to perform functional and integration testing as well as scalability and robustness checks on the code base, these deployments rarely get extended beyond one or maybe a few physical servers. In the case of edge architectures it is crucial to check functionality that is designed to overcome the geographical distribution of the infrastructure, especially in the circumstance where the configurations of the architectural models are fundamentally different. In order to ensure stable and trustable outcomes it is recommended to look into the best practices of the scientific community to find the most robust solution. One common standard practice is the artifact review and badging approach.

Testing is as much an art form as it is a precise engineering process. Testing code on lower levels, such as unit tests or checking responses of components through API tests, is straightforward. This allows frameworks to be created that support running an automated unit test suite that addresses requirements such as repeatability, replicability and reproducibility. Testing the integrated systems to emulate the configuration and circumstances of production environments can be quite challenging. The diagram below describes the general process that is executed when performing experimental campaigns. This process, that is applied in the field of research, can also be utilized to help build new components and solutions that fit the requirements of edge computing use cases even though some of the steps still need more tools to perform all checks as if they were simple unit tests.



The first seemingly trivial step describing the acquisition of resources from a testbed is not specific to edge computing scenarios. The assigned resources (e.g., compute, storage, network) represent the physical infrastructure that will be used to conduct the evaluation.

The second phase is more difficult. It incorporates multiple sub steps to prepare the physical infrastructure as well as the deployment of the system under test (SUT). As edge environments can be very complex, they also need to be tested for their ability to be prepared for circumstances such as an unreliable network connection. Therefore, having a deployment tool that supports a declarative approach is preferred to specify the characteristics of the infrastructure such as latency, throughput and network packet loss ratio to emulate the targeted real life scenario and circumstances.

Once the deployment plan has been created and the resources have been selected, it needs to be confirmed that the infrastructure is configured correctly during the pre-deployment phase before installing the applications and services on top. This is especially true in edge architectures where resources must be available over complex networking topologies. For instance, profile attributes may have all been set correctly, but are all the resources reachable, in good health, and can communicate to each other as expected? The checks can be as simple as using the ping command bi-directionally, verifying specific network ports to be open and so forth. The purpose of this procedure is to ensure that the deployment step will be completed successfully and result in a test environment that is aligned with the requirements and plans. The complexity of edge architectures often demands a granular and robust pre-deployment validation framework.

Now that the testbed is prepared and tested, the next step is to deploy the software applications on the infrastructure. For systems built on environments such as OpenStack and Kubernetes services, frameworks like Kolla, TripleO, Kubespray or Airship are available as starting points. Be aware that the majority of these tools are designed with the limitations of one datacenter as their scope, which means that there is an assumption that the environment can scale further during operation, while edge infrastructures are geographically distributed and often have limited resources in the remote nodes. In addition, the configuration options are significantly different among the different models. As part of testing edge architectures, the deployment tools need to be validated to identify the ones that can be adapted and reused for these scenarios. To ensure the success of testing, the installation itself needs to be verified, for instance, checking the services to ensure they were installed and configured correctly. This operation should preferably be a functionality of the deployment tool.

When all the preparations are done, the next step is benchmarking the entire integrated framework. Benchmarking is often defined as performance testing, but here it applies to a broader scope that includes integration and functional testing as well. It is also important to note that the test suites can be heavily dependent on the use case, so they need to be fine tuned for the architecture model being used. While a few tools exist to perform network traffic shaping and fault injections, the challenge lies more in the identification of values that are representative to the aforementioned edge use cases.

Building an edge infrastructure consists of various well known components that were not implemented specifically for edge use cases originally. Because of that, there are situations where there will be a need to test basic functionality in these environments as well to make sure they work as expected in other scenarios. Example functions include:

- create/delete a resource (user, flavor, image, etc); scope: one or more edge sites
- list instances (VM, container); scope: an edge site or ‘single pane of glass’ dashboard
- create resources for cross-data-center networks

Further testing of the edge infrastructure needs to take the choice of architectural model into consideration:

- Using OpenStack in the centralized control plane model depends on the distributed virtual router (DVR) feature of the OpenStack Network Connectivity as a Service (Neutron) component.
- The behavior of the edge data centers in case of a network connection loss might be different based on the architectural models. In some cases, the decision might be to choose to configure the system to keep the instances running while in other cases, the right approach would be to destroy the workloads in case the site becomes isolated. In addition to these considerations, the expectations on functions such as auto-scaling will also be different due to possible resource constraints, which need to be reflected in the test suites as well.

The final two steps are trivial. The test results need to be collected and evaluated, before returning the SUT infrastructure to its original state.

Tools such as Enos, Enos-Kubernetes and enoslib are available in the experiment-driven research community to evaluate OpenStack and Kubernetes in a distributed environment over Wide Area Network (WAN) connection. They can be extended or leveraged as examples of solutions that can be used to perform the above described process to evaluate some of the architecture options for edge. Further components are needed to ensure the ability to test more complex environments where growing numbers of building blocks are integrated with each other.

Conclusion

Edge computing is highly dependent on lessons learned and solutions implemented in the cloud. Even if the majority of building blocks are available to create an environment that fulfills most requirements, many of these components need fine tuning or API extensions to provide a more optimized and fit for purpose solution. Deployment and testing requirements are further highlighted for these new architectural considerations, and therefore existing solutions need to be enhanced, customized and in some cases designed and implemented from scratch.

The real challenge lies in efficient and thorough testing of the new concepts and evolving architecture models. New test cases need to be identified along with values that are representative to typical circumstances and system failures. Testing can help with both enhancing architectural considerations as well as identifying shortcomings of different solutions.

As can be seen from these discussions, edge computing related innovation and software evolution is still very much in its early stages. Yes, there are systems running in production that resemble at least some of the considerations—uCPE or vRAN deployments, for example. The architecture models discussed here cover the majority of the use cases, however, they still need additional efforts to detail the required functionality to go beyond the basics, outline further preferable solutions and document best practices.

This is the perfect time for groups in the IT industry, both open groups and semi-open or closed consortiums, as well as standardization bodies, to collaborate on taking the next steps for architecture design and testing in order to be able to address the needs of the various edge computing use cases. Considering the high level of integration needed, it is crucial that the subject matter experts of the various components start to contribute to a common effort.

Authors

- Beth Cohen, Distinguished Member of Technical Staff, Verizon
- Gergely Csatári, Senior Open Source Specialist, Nokia
- Shuquan Huang, Technical Director, 99Cloud
- Bruce Jones, StarlingX Architect & Program Manager, Intel Corp.
- Adrien Lebre, Professor in Computer Science, IMT Atlantique / Inria / LS2N
- David Paterson, Sr. Principal Software Engineer, Dell Technologies
- Ildikó Váncsa, Ecosystem Technical Lead, OpenStack Foundation