Intel

RAN Intelligence Enabling

CORPORATE PARTICIPANTS

Lilian Veras
Moderator

Michael Recchia
Red Hat – Global Telco Solutions Architect

Federico Rossi
Red Hat – Senior Solutions Architect

Hassnaa Moustafa
Intel – Principal Engineer

Sunku Ranganath
Intel – Customer Solutions Architect

PRESENTATION

Lilian Veras

Welcome, everyone, to the Intel Network Builders webinar program. Thank you for taking the time to join us today for our presentation titled, “RAN Intelligence Enabling”.

Before we get started, I want to point out some of the features of the BrightTALK tool that may improve your experience. There's a questions tab below your viewer. I encourage our live audience to please ask questions at any time. Our presenters will hold answering them until the end of the presentation. Below your viewing screen, you will also find an Attachments tab with additional documentation and reference materials.

Finally, at the end of the presentation, please take the time to provide feedback using the Rating tab. We value your thoughts and we will use the information to improve our future webinars.

Intel Network Builders webinar series takes place live twice a month, so check the channel to see what's upcoming, and access our growing library of recorded content. In addition to the resources you see here from our partners, we also offer a comprehensive NFV and SDN training program through Intel Network Builders University. You can find the link to this program in the Attachments tab, as well as a link to the Intel Network Builders newsletter.

Intel Network business partners have been working to accelerate network innovation by optimizing their solutions on Intel technologies. These industry leaders are recognized in our Winners’ Circle program, and Red Hat is a Titanium Partner. Learn more about our INB Winners’ Circle program by clicking on the link in the Attachments tab.

Today we're pleased to welcome Michael Recchia and Federico Rossi from Red Hat, and Hassnaa Moustafa and Sunku Ranganath from Intel.

Michael is a Global Solutions Architect at Red Hat and focuses on the strategy, development, and execution in Open RAN, including development of the first near-RT RIC and several xApps, including Dual Connectivity, SON ANR, Spectral Awareness, Anomaly Detection, and Traffic Steering.

Federico Rossi is a Senior Solutions Architect at Red Hat. He's not in the talk today, but he contributed with a demo we will watch later.

Hassnaa Moustafa is a Principal Engineer at Intel Corporation, currently working on Edge Computing and AI solutions across IoT segments and Network Edge Infrastructure. Previously at Intel, Hassnaa led Car-to-Cloud solutions for connected autonomous vehicles, connectivity technologies across IoT segments. Before joining Intel, Hassnaa was a senior R&D engineer at Orange in France,

where she contributed to cost-efficient wireless network solutions for EMEA, and led engineering efforts on video and multimedia services optimization over wireless networks.

Sunku Ranganath is a Solutions Architect for Edge Compute at Intel, with a focus on enabling solutions for the telecom domain, including designing, building, integrating, and benchmarking NFV-based reference architectures. Sunku is an active contributor to multiple open-source initiatives, served as the maintainer for CNCF Service Mesh Performance and CollectD Projects. He's an invited speaker, contributor to standard bodies, and a senior member of IEEE.

Welcome, Michael, Federico, Hassnaa, and Sunku, and thank you again for joining us today. I will hand over to Hassnaa to start off. Thank you.

Hassnaa Moustafa

Thank you, Lilian, and thank you, everyone, and thanks to my co-speakers today. The topic is RAN Intelligence Enabling, and we will discuss with you the joint efforts Intel and Red Hat are doing in this area.

Before I start speaking about RAN intelligence, I would like to talk about the Edge, and Intel's story in the Edge, as we see tight intercepts now between the RAN, and RAN intelligence specifically, and the world of IoT and Edge Computing. We're seeing a convergence happening. Now we cannot think of the RAN and RAN intelligence in a standalone manner without giving the story of the Edge, and how RAN and RAN intelligence are interacting with the different Edge deployments.

So I'll start by this picture, giving you an idea of how we see the transformation in the Edge infrastructure, and how we see the Edge as a composition of cloud-native platforms, as if we're moving the cloud resources to the Edge, and building an Edge infrastructure in a cloud-native fashion to mimic the cloud environment, but making it at the Edge.

As we see in this picture, we look at the Edge, and Intel's story for the Edge considers diverse Edge locations, diverse types of Edge deployments, which can start with an on-premise Edge, and mainly triggered by its IoT broad segments and broad types of applications that can be industrial IoT, IoT for retail, IoT for health and medical applications. We find that all these types of applications, of course, trigger the need for the Edge, but looks more for an on-premise Edge deployment, which goes hand in hand with Private 5G here. There are more types of Edge deployments that can go to the Edge infrastructure, network infrastructure, access Edge, and even Regional Data Center Edge. We see also Edge deployments as a core network, collocated sometimes with 5G Core, with the UPDF, and in cloud data center for the telco. Intel plays a role in all these types of Edge deployments, bringing different enablers for best performance, for optimization.

This type of diversity for Edge deployment is very good to meet different needs for latency. As you see here in this picture, the more sensitive the latency, the more the trend is towards an on-premise Edge deployment. There are also other factors such as security. Sometimes there are certain types of applications that require the data to be really local and doesn't leave the on-premise infrastructure. There are other types of applications that are more relaxed in terms of latency. Latency can go up to 60 milliseconds, and it needs to be in a hierarchical point of infrastructure to serve multiple users in different areas, and here we go more to the Network Infrastructure Edge and Regional Data Center Edge. That's an opportunity for different players to play a role at different Edge locations. We see also other opportunities in terms of revenue tied with Private 5G, Private Wireless revenue from hardware and software according to the different analyses. It goes up to $29 billion expectation. We see also a trend for 75% of the data in a few years from now will be processed and will take place at the Edge. All this helps us to move forward and try to go fast, and enable intelligent solutions at the Edge, enable intelligent RAN that can serve different Edge services with the best quality of service.

What are the key technology inflections here that we are observing and working on for such an Edge ecosystem? Cloud-native software, I spoke about it. We need to mimic the cloud environment, but bring it to the Edge, and make it compatible with the Edge requirements. So we need cloud-native software.

Connectivity, of course, is needed. 5G is our main connectivity here, but it's not limited to 5G. 5G coexists with other technologies, Wi-Fi or others, and multi-access is highly observed by different applications here at the Edge.

AI, of course, that's a must-have, and as we speak today, and as you will see, we're not looking at AI only for the services. It's not only AI to provide intelligent applications, but AI to enable the RAN. If the RAN is intelligent, taking intelligent decisions, automated decisions, it will serve the network infrastructure. It will serve also the services on top. So today, we will show more on how Intel and Red Hat are considering all those points.

What do we do at Intel to help accelerate and provide optimization on Intel hardware for this cloud-native, and we call it also Edge-native environment, which needs to happen at the Edge, is we offer a software product, Smart Edge. Some of you've heard about it before, whether through working with Intel in different projects or through different meetups we had before. So Smart Edge is a software product that we offer, and it's a Kubernetes-certified product. It's certified by the CNCF as a Kubernetes-certified product, and it's an Edge-native for us. We are bringing the cloud resources, the microservices style, the modular approach, and we are bringing it to the Edge, and providing different optimizations on Intel hardware. We consider it as an Edge-native, adopting the cloud approach and making it work at the Edge. So it's Kubernetes-certified and it follows the cloud-native approach in an Edge deployment. It enables deployment management of container-based workloads, microservices, with resiliency, with security at the Edge. It enables diverse workloads to run together in a coexistent way at the Edge. Most of our workloads that we consider for the Edge services are AI workloads, AI related, media related for AI-based services for media services, and also software-defined functions for the network. You will see in the coming part of the presentation that Smart Edge enables a RAN workload, RAN CNFs, and 5G Core CNFs even, to coexist with other AI services on the same Edge Platform or separate Edge Platforms based on the deployment, and it offers a revalidated blueprint for this existing workload, and with optimization on IA.

We have different reference implementations we developed for working with Smart Edge to show our partners how they can use Smart Edge and build their solutions.

How do we offer it? We offer it in several forms. A form we call "Develop with us" for developers, for IoT developers, for Edge solution developers, who want to build an application to run on an Edge Platform, and to be an Edge-native application, microservices-based. So we offer a Smart Edge flavor for developers, and this observes building any application for IoT, for Edge service, or even for a RAN, building a xApp, onboarding it, and testing it with Smart Edge. This is an opportunity as well, and you will see more in the rest of our discussion today.

We have another flavor that is built with us, which targets ISVs, which target Edge solution providers who want to build an Edge solution for Private 5G or for the enterprise, like the SASE type of deployment. We offer this version for builders, and it's a royalty-free license. It has Intel basic support and is available on our GitHub.

For the version for developers, I mentioned earlier it has the Apache 2.0 license, and it has community support.

We go a step ahead as there is a version that is "Buy from us" for those who are looking for a turnkey deployment for a plug-and-play solution. In this case, as they go and they take this commercial version and deploy it, and it has premium support, and it's a paid license. So we cover the different needs that we see from our customers through the different flavors we offer. This was like the Smart Edge, our product that we bring for the Edge to accelerate the ecosystem, to enable our partners to build Edge solutions, to do different Edge deployments.

How it looks like from an architectural point of view, if we look at this picture from a high-level point of view, all starts with the Intel Edge Platform, and on top of that, we find the Smart Edge, which is built on top of Kubernetes as I mentioned, and it's optimized on IA. It has different building blocks, different microservices, as we see in this layer. We have building blocks to enable optimization on our platforms using different Intel IPs, Intel hardware IPs, like the eASIC for the vRAN workload acceleration, FPGAs, QAT for IPsec for crypto acceleration, the different NICs products of Intel. A GPU even is coming soon, the recent GPU of Intel. So all those are cloud-native plugins we offer with Smart Edge, helping any application developer or any solution builders on top to use our IPs with ease and

in a handy way, and get the optimization needed. We offer different building blocks for resources management, security, local distributed storage, and more details will come in our talk today.

That's the layer between the platform and the service. If we go up an additional layer, we have additional microservices or a set of microservices. If we look at RAN, we offer, for example, with Smart Edge any reference for a near-real-time RIC implementation. Currently, there is one available from the open source, from Open Network Foundation, with xApps as a demonstration of how this can come as a reference with Smart Edge. We work with third-party 5G Core vendors and third-party RAN vendors who bring their CNFs for the RAN and for the 5G Core, and it is working and interoperable with Smart Edge. So these are microservices for the RAN we offer. After that, we offer other microservices for AI, different components of Intel OpenVINO for Edge AI inference acceleration. Edge Insights for Industrial is another set of microservices to optimize ingestion and inference, and even display and share insights for analytics applications for the industrial environment, and other libraries from Intel for media and real-time communication, and in microservices forms.

This wide set of microservices is to build IT services with optimization on IA, and we offer reference implementations showing how all the microservices can be utilized to build a solution. Reference implementations can be downloaded from Intel DevCatalog. It can be used and tested and evaluated.

Here I'll give just a quick highlight on what we do here for RAN intelligence. I spoke about reference solutions and microservices for the RAN. We started AI capabilities by focusing on AI for services to build AI-based services. We moved a step ahead for more than a year now, and we are expanding the Edge AI capabilities to RAN intelligence. Why'd we do it? We'll come to this in the rest of the talk because, as I mentioned, we cannot separate the RAN from the Edge story, and from the Edge AI services. We have convergence in terms of workload between the network workload and the services workload. So we brought different microservices to enable RAN intelligence. As I mentioned, near-RT RIC from the ONF SD-RAN project, it's tested and interoperable with Smart Edge. We brought even xApps from Intel Labs Research and we tested them with SD-RAN RIC, and these xApps, they have the AI different neural network models optimized with OpenVINO optimization, and is meeting real-time needs for the 5G needs.

So, all those are in cloud-native form, microservices form existing as part of Smart Edge building blocks, or microservices to enable RAN intelligence. And a reference solution is built, you can download it through Intel Dev Catalog and test it and evaluate it.

So, by here, I'll stop, and I'll hand it off to my colleague, Sunku. He will continue the discussion with you.

Sunku Ranganath

Yes, thanks, Hassnaa. For the interest of time, I'll be quick.

So, here, we could see one of the reference solutions Hassnaa talked about. Near-real-time RIC is one of the important components in the O-RAN specification that connects CU and DU, and helps take decisions in real-time. When it comes to near-real-time RIC, the underlying infrastructure needs to play an important role in terms of being resilient, secure, sensitive to requirements around latency, and in general overall performance, and provide infrastructure components to help achieve these aspects within a certain SLA.

Smart Edge provides a set of building blocks to help accomplish them via these building blocks, as you can see here. It leverages Kubernetes and surrounding open-source components with respect to CNI, and some of the microservices from Kubernetes. We built a reference solution with Smart Edge's Developer Experience Kit, and ONF's SD-RAN software for evaluation of RICs and xApps on Intel architecture.

The reference solution provides automated deployment of end-to-end platform for near-RT RIC. It leverages multiple data plane CNIs for optimum performance. It utilizes RAN simulator and near-real-time RIC from SD-RAN. And of course, provides a set of xApps from Intel Labs research for anyone to deploy and understand the intricacies involved in building near-real-time RIC.

And ultimately, it helps to validate various performance KPIs. It helps engage with various partners by acting as a testbed for trials in pre-commercial deployments. It can help communication service providers and RIC vendors to build their own RIC.

Here are some of the key building blocks from Smart Edge that we can utilize based on the RIC workload and the xApps that we are deploying. It covers broadly across a different set of domains from a container orchestration service with the certified CNCF Kubernetes distribution optimized for compute and I/O, versus platform provisioning services, an automated way of installing and configuring the firmware, all the drivers and software stack on top. And 5G RAN or core services domain with respect to deterministic I/O, required acceleration for these components. SASE services in general for Secure Access Service Edge. And also, a set of microservices and underlying hardware components for zero-trust security with respect to platform attestation, integrity, our trusted execution, et cetera.

We're looking into also leveraging service mesh across L3 and L7 planes as well. And if you're deploying multi-tenancy-based services, you need a level of resource isolation and security from the hardware to the software services. So, you have a set of technologies on the hardware like RDT, Kubernetes namespace, control SGX, et cetera that you can leverage.

And there are Green Edge Services that help conserve power. Data plane networking services around various CNIs, acceleration, DPDK, SR-IOV, Nodus, et cetera.

Accelerator services, when you look at end-to-end deployment with respect to FPGA or FEC acceleration, and dynamic device profiles and NIC, et cetera.

And then services related to storage and observability with telemetry, logging. And Hassnaa talked about video analytics services. And of course, a set of operating system services in order to provide the real-time performance capabilities.

So, one of the important building blocks that we look at leveraging for this talk is around telemetry. From a platform perspective, there are a lot of metrics and telemetry components that you can leverage in order to provide a near-real-time solution, and help achieve a deterministic performance for a near-real-time RIC. Some of the aspects indicated here around Intel Infrastructure Management technologies with RDT, Run Sure Technology, Trusted Execution, Rapid Storage. And of course, differentiating metrics with respect to PMU, Performance Monitoring Unit, NIC metrics, et cetera.

So, when you look at leveraging some of these, you can leverage open-source components like Collectd, or Telegraf, and integrate these metrics. And Smart Edge provides you easy and simple interfaces for you to consume these metrics.

Now, the question is, how do I use them? How does it apply to my RIC-based deployment? And this is where Kubernetes-based Telemetry Aware Scheduler comes into play. We'll look at a simple demo towards the end.

However, the important thing to understand here is as you collect these metrics from Collectd or Telegraf, the control plane needs a way to understand what's going on in the individual nodes across the hardware consumption, hardware utilization, et cetera, and be able to make decisions at runtime to adjust resources or move around workloads based on the service level agreements that are being set for near-real-time RIC.

Some of the scenarios we are looking at are noisy neighbor solutions, QoS and QoE fine-tuning, platform resilience, real-time resource management, et cetera.

So, Telemetry Aware Scheduler provides a simple way of consuming metrics from your Telegraf into Prometheus and from Prometheus into control plane – Kubernetes control plane using TAS-based components in the control plane. And based on the metrics that are set by the user, for example here, there's a health_metric2 versus 5, the 5 being green, 2 being red. So, the set of parts would continue to run on Node 1 with the health_metric5 versus Node 2 with health_metric2.

So, you can mix and match and change your SLA, and change your rules in the control plane, leveraging the real-time metrics happening on individual nodes.

So, look at the demo later to get into more details from a RIC perspective.

So, with that, I'll pass it on to Mike.

Michael Recchia

Hello, everybody. I'm Michael Recchia with Red Hat, Global Telco Solutions Architect. I spent most of my career with AT&T where I built with Nokia, we did a co-create and built one of the first near-real-time RICs in the industry.

So, I'd like to talk today about application of the Intel Building Blocks to various components that we're using to support the RIC.

So, here, I'm just showing the Intel Optimization Building Blocks Library, and what we're doing with it. We're planning to do studies, and we've done some preliminary studies where we've applied it to the OpenShift container platform itself, to the RIC software modules, and to use cases involving the RIC, basically calling those end-to-end RAN use cases. The idea here being that, in the end, we have an Optimized OCP. We have an Optimized RIC, and we have, in essence, an optimized end-to-end use case.

And again, the categories are these three. OpenShift Container Platform Proper, looking at some of the native internal capabilities of OCP, and how the building blocks can help us improve performance.

And then the near-real-time RIC Proper, which involves several modules within the near-real-time RIC itself. Some of the more important modules being things like the E2 Termination, E2 Manager, and an xApp Subscription Manager.

And then the use case itself, which is looking at things like the network layer flow, like packet flow and packet loss. And some of the use case objects like SDN and RIC modules and so on.

This is a picture of the near-real-time RIC on some of the major software components of the near-real-time RIC. And you can see these little yellow BB boxes. The intent is, over time, we would be applying those building blocks to those major architecture components of the RIC. And there may be an opportunity, at some point, to use some of the Red Hat middleware as part of the near-real-time RIC infrastructure, not to create our own near-real-time RIC, but to do some optimizations in addition to the building blocks. So, you can see in the red, some of the middleware that we're thinking about experimenting with for some of the major components of the near-real-time RIC.

And so, the idea here being what are the major processes of the near-real-time RIC? And how do the building blocks help us improve the performance?

So, these are the three vectors to optimization – I call them Pathways to Optimization – that we're looking at, at Red Hat. One of them being the Intel Optimization Building Blocks or the IOBB. That includes RHEL and OCP services, the near-real-time RIC and xApps, maybe the non-real-time RIC and rApps, and then SMO, of course. And then I talked about improving some of the functions within the RIC with some Red Hat middleware. And then looking at how we do workload distribution across a Kubernetes cluster, or an OCP cluster based on some criteria.

And so, one of the things that we're actually very interested in doing is doing workload distribution based on power consumption. And in fact, when you see the demo, you'll see some reference to power consumption for some of the processes and how decisions are made to schedule processes based on power consumption.

And then we're also looking at communicating SLA requirements to the cloud layer for end-to-end cellular use cases. For example, for network slicing, where you may have multiple subnets, and you want to be able to communicate the performance requirements to each of the cloud – cloud supporting each of the subnets within a slice.

So, one of the things we're doing here in our lab is we are building some use cases. And one of the first use cases that we're doing with the RIC, what we call Anomaly Detection with Traffic Steering. And so, we're starting with the O-RAN-SC open source RIC. Actually, that's based on the first RIC that we did with Nokia, that served as the seed code for the O-RAN-SC RIC. And we're going to be doing anomaly detection, followed by some traffic steering. And another name for this is QoE Optimization, where we're basically doing some machine learning or deep learning and looking at the RAN behavior over time, and then making inferences about either congestion or interference, or both that might be occurring in the RAN and then proactively moving UE via traffic steering as opposed to waiting for the network itself to do the cutover of the UE. So, it's really maximizing the QoE of the UE, in essence.

And so, some of the prerequisites for that are this so-called Far Edge Footprint. The telcos, in particular, are using these x86-based chassis that are meant to be deployed in a mobile context that can be deployed anywhere, basically, including outside. So, one of the goals that we have is to use that footprint in the lab and then do the use cases that we're talking about on those form factors.

And here, you can see what that might look like initially. On the left side, you can see a gNodeB emulation, with some E2 sims that we're using in the lab. Eventually, we'll be migrating to an actual gNodeB with an actual E2 client. But initially, we'll be running our use cases using E2 simulators. And we'll be simulating an environment where there's some anomalies going on in the RAN, and then there will be xApps that will detect the anomalies, and then take some actions to mitigate.

So, you can see there's an AD xApp in the middle sitting on top of a near-real-time RIC. And you can see at the bottom, there's some major near-real-time RIC processes like the E2 termination module. And of course, the near-real-time RIC is running on an OpenShift container platform. And the building blocks are being applied in all of these areas.

So, they'll be applied to the RIC and then to the xApps and to OCP, and basically, to the end-to-end use case.

On the right is basically just a blow-up of what the near-real-time RIC looks like. So, you see there's a dotted line encapsulation of some of the major modules within the near-real-time RIC. And you can see the traffic steering xApp, there's an interference xApp, and a congestion xApp. Those are the anomaly detectors, and then the traffic steering xApp then takes action based on that.

And as I said, over time, on the left, we'll be migrating to an actual gNodeB. What that means is O-CUs and O-DUs with actual E2 clients running on them. The building blocks will be applied there as well. In the middle is your near-real-time, RIC. And you can see, again, the little yellow boxes with the BB inside, those are the Intel Building Blocks. They'll be applied to the near-real-time RIC. At the bottom you see OCP itself, the building blocks will be applied there.

So, in the end, when you run the end-to-end use case, or you do the interference or congestion detection, and then you do the traffic steering away from the anomaly, if you've applied these Intel Building Blocks properly, you should get an optimized end-to-end use case, because it is essentially being applied everywhere. So, there's resource optimization on the gNodeB. There's resource optimization on the RIC itself. And then later on, perhaps, we move up the stack going to the non-real-time RIC and the SMO.

Ultimately, one of the goals that we have at Red Hat is to build blueprints for major 5G use cases. And so, one of the things that we're looking at doing is optimizing these blueprints for those use cases and incorporating the right vendors that will help us do that optimization.

So, every blueprint will have some kind of value. So, revenue value, and an OpEx, CapEx reduction value. And OCP will be the major infrastructure component of the blueprint. And the goal is to make Intel Optimization Building Blocks a part of that blueprint. And so, they would be applied horizontally as well as vertically, meaning that they're applied across the service or the use case end-to-end. So, whatever the blueprint supports in terms of the 5G use cases, they're applied across the use cases. And then vertically, they're applied to each of the technologies that sit inside the blueprint.

So, a typical blueprint for a RIC application will have xApps, you'll see... we'll have CUs and DUs, and we'll have, perhaps, an integrated MEC. And we'll have, perhaps, some OSS stubs from different vendors and some tests – maybe some test and turn-up stubs, and of course, OCP. So, the bottom line is that the goal is to produce these blueprints for the major 5G use cases, and the blueprints are optimized for those use cases.

One of the things we're looking at for down the road is how do we signal to the cloud layer – especially for network slicing – how do we signal to the cloud layer so that we can be true to the end-to-end SLA? Because an MNO or a telco is going to sell a slice to an MVNO. The MVNO is going to be contracted for a particular SLA, so the telco has to live up to that SLA. So, how do you make sure that you can provide that SLA given that each of the subnets within a slice could have their own unique cloud infrastructures?

So, one of the things we're looking at, or experimenting with, is passing down some parameters from the network layer, the network slicing layer to the cloud layer. So, you can see three of them in particular here would be service attributes, or SA, and maybe a

workload optimization matrix, or just a stake in the ground and saying, “We believe that the workload should be distributed across the cluster in this particular manner”. And then perhaps an IOBB matrix that basically says, “Here are the Intel Building Blocks that we think should be applied at the cloud layer, so that we can provide – we can be true to the SLA”.

So, this is experimental. This is something that we think we can do. But of course, there's a lot of work to be done. And hopefully, at some point, the goal, in the end, is to make this a CR, basically update the 3GPP spec and maybe the O-RAN spec for this kind of use case.

So, this – Sunku talked about Telemetry Aware Scheduling. The demo was produced by Federico Rossi who is a Senior Solutions Architect at Red Hat. And basically, the demo is going to show how we can use the scheduler, a TAS, and the secondary scheduler to optimize the distribution of workloads across the cluster.

In this particular demo, it's really what happens to scheduling when you apply more power to a particular pod than that pod can handle. And then there's a redistribution of workloads based on that.

So, the demo itself will have – I think they call it Node 7, 8, and 9. We're calling them Node 1, 2, and 3 in this slide. But I'm going to launch the demo now and you'll see how the TAS provides an opportunity for power-related redistribution of processes based on power consumption. So, let's go to the demo.

Federico Rossi

Hello, everyone, welcome. In this short demo, we will demonstrate how Intel Building Blocks for telemetry can be used on an OpenShift cluster to distribute workload based on nodes power usage. Let's get started.

On the bottom terminal, we have the output of the TAS, the Telemetry Aware scheduler. In this demo, we have an OpenShift cluster of three nodes, the nodes are called Node 7, Node 8, and Node 9. The nodes have a master and worker role. We will deploy a TAS policy that is defined as follows.

The policy is composed by strategies and rules. In this simple policy, we are using a “scheduleonmetric” that is mandatory for the policy. And it uses the metric name, Collectd Package Power, basically, monitoring the power of the node. And it says if less than 30, as a value, you can schedule. We have a deschedule strategy that says if it's greater than 34 the pod will be descheduled, it will be evicted from the node. And then we have an additional strategy that says don't schedule whenever the power is greater than 32.

This is a simple policy. But you can create more advanced rules and combine them as well with different metrics and also use logical operators such as and/or. For example, you can say, when power – schedule if power is less than 30, and the temperature on the node is within a certain value. So, as you can see, you have a lot of flexibility to control this policy.

We will now apply the policy. And we can see that the TAS is registering the policy and is now starting to evaluate it. We'll see in a few seconds that it will start pulling the metrics with the corresponding value on the node. There you go.

Here we have Node 8, 9, and 7 and the correspondent current value. So, we're now going to deploy a deployment that basically deploys just an NGINX pod for testing. And this deployment will use the TAS as a policy and a scheduler for the scheduling.

As you can see, as soon as we create the deployment, the TAS has received a filter request from the secondary scheduler. And has replied the available nodes. Since the power is below the values that are defined on the policy, the filter nodes are all returned on the output. That means the scheduler could pick between Node 9, Node 8, or Node 7.

So, let's have a look where this pod ends up. As you can see, here we have a demo app, and it's on Node 7. We will now load using stress-ng one of the nodes. In this case, let's say Node 9. We deploy a pod that has stress-ng and we use a node selector to make sure that that pod ends up on the Node 9. Here we're seeing it is running on Node 9 and we can now log in on the pod and start generating computational stress to the CPU.

What we'll see happening is that Node 9 will start violating the policy that we defined. Look at the policy again right here. It says don't schedule if the power is greater than 32, right here. Let's wait a few seconds. The TAS is actually reconciling the policy every 10 seconds in this case, but the scraper for biometrics runs every 30 seconds right now. Of course, that can be changed as needed.

Okay. Here we go. As you can see, Node 9 is violating the policy that we have defined. At this point, if it tried to scale the deployment to two pods, two replicas, it will say then when it tried to schedule the pod is sending the request to the TAS and the TAS will reply that only Node 7 and Node 8 are available.

There you go. So, here I filter it. Filter nodes, where – it's right here, filter request received, filter nodes for power policy, Node 7, and Node 8. So, Node 9, it was not anymore available, because it's currently violating the policy. It is using too much power at the moment. And actually, that's, as you can see right here, 75 compared to 30, and 25 of the other node.

At this point, we're going to demonstrate how the node is descheduled automatically. So, let's see where the node ends up. Not sure why this is here. Demo application, they are both on Node 7. That means if I generate a load on Node 7 right now and it reached more than 34, it will be descheduled and Node 8 or Node 9 will be selected in this case. So, let's do that.

So, let's stop our stress-ng on Node 9, we're going to delete this one.

Okay.

Just give it a few seconds. The power here is still high, let's wait for that to reset.

While it's doing that, I'm going to create the stress pod on Node 7. In the meantime, you can see that Node 9, this is going down. There you go. Now, we're back to normal right here. It's not – Node 9 is not violating the policy anymore. So, we deployed stress-ng on node seven. Logging onto the pod.

Okay, running now, there you go. And in this case, instead of logs of the TAS, I'm going to position myself on the logs of the descheduler. OpenShift. Kube. Descheduler. Operator. Logs. Cluster. There we go. The descheduler, in this case, runs every 100 seconds. So, let's start generating the load. And here... We have to wait that the policy... yes, let's see, log the TAS. You see the load when the Node 7 goes in violation.

I remember that I have the deschedule strategy that says when it's greater than 34. Since I have both pods for demo app on Node 7, when Node 7 gets bigger than 34, it's going to evict the pods from the node and distribute it across the other ones. There you go.

Now, Node 7 is violating the policy, I can go back to the descheduler, and here you can see we were able to catch it in time. It's filtering, processing node, right here. Evicted pod, demo app, reason “NodeAffinity”. And number of evicted pods, total evicted, two, and as you can see that the pods were moved to Node 9 right here. So you got pods. There you go. Node 9.

And this is it for demonstrating distribution of workload based on power usage leveraging Intel Building Blocks for telemetry using the Telemetry Aware Scheduler on OpenShift.

Thank you for listening in and enjoy.

Michael Recchia

Okay. Well, thank you, everybody, for participating today. Just to close on that little note on that one. Of course, one of the objectives is to be able to distribute workloads based on power consumption. So, if you do the m-choose-n algorithm, if you will, of the workloads you can, in theory, optimize the power consumption in the sense that you minimize it. And that's one of the goals we have is to make this a green effort, so that we can use the building blocks to do that. But that ends the discussions and now we'll go to questions.

Lilian Veras

Thank you, Mike, Sunku, Hassnaa – and Federico also for the demo – sharing such great information with us. We do have a few questions that have come in while you were presenting. So, let's get started on those.

Question number one. “How do you see AI improving both the network performance and the service? And how Intel and RH, play a joint role, Red Hat, play a joint role in this area?”

Hassnaa Moustafa

Thank you, Lilian, and a great point here. We discussed different points during this discussion how AI can help services, and can help RAN intelligence of course. And Mike spoke a lot here and we saw the demo.

So, first, think of it as an Edge story and the transformation of the cloud resources to the Edge. We have an infrastructure at the Edge now enabling connectivity and enabling Edge services. Connectivity here, it means 5G, it means multi-access, it means RAN, it means 5G Core. And there is AI services at the Edge also for IoT, for Edge Computing, for MEC.

If we look at AI separately, we will not do efficient use of our resources, software components, as well as hardware. That's why we see the workload conversions, we see the 5G workload converging with the IoT workload, converging with the MEC workload. So, AI itself is – we see it as horizontal, and that's how the world is seeing as well, enabling both – enabling optimization for the services and enabling optimization for RAN. That's the logical view.

But how we do it ourselves, how we enable it in that way, we offer software capabilities like OpenVINO optimization as an example for inference optimization. We use it in a dual mode. We use it to accelerate and optimize inference for any AI service. And to optimize the inference also for neural network models for an xApp use for RAN intelligence.

So, AI here is – that's an example when I say OpenVINO is having a dual role to help optimization for IoT service, as well as for RAN intelligence services, which is xApp that Mike spoke about. And it's a cycle, meaning when the network – when AI makes the network intelligent, the network will make a better decision or quality of service that will help the service... the end application quality of service as well.

So, it's a cycle also where when we make the network intelligent, it will help better the services, and make it with more quality of service.

So, I don't know, Mike, if you have something to add here on how our collaboration helps this convergent view for AI.

Michael Recchia

Well, I mentioned one of them. One of them is workload distribution. What's the optimal way of doing that based on some criteria? For example, power consumption would be one of them. So, AI could definitely help there. So, what's the optimal algorithm for distribution of workloads based on power consumption? And then we can use the scheduler, the building blocks to do that. So, that's one area.

The use case itself that I presented was QoE optimization, or proactive Quality of Experience optimization, meaning using the RIC and associated xApps to do that. In other words, to do forecasting, what's happening on the RAN, what will happen if we don't take some action. And so, that's the use case, and then applying the building blocks to the use case to optimize that functionality is another area.

And then integration. One of the things I've been pushing and one of the things we did when I was at AT&T with Nokia was integrate the MEC with the RIC so that you could optimize the RAN for particular services that are running on the user plane. So, there's sort of this mapping function between the microservices running on the MEC and the xApps running on the RIC so that you have this very tightly coupled optimization. So, for a given service, you're tightly coupled to a particular optimization.

So, those are three areas right off the bat, but there's many more. So, we're at the beginning stages of this study. Expect more as we collaborate, there'll be many more areas where we will find—

Sunku Ranganath

Yes, just to add on to what Mike mentioned. In the example we took from a telemetry standpoint, you can look at some of the closed-loop automation solutions. The sheer amount of data and metrics one would get across from applications and infrastructure from your hardware, it's literally very difficult or nearly impossible for an individual to look at these metrics or run simple heuristics and take decisions in real-time. So, AI plays a huge role in closing the loop, so to say, across your individual node, or your cluster, or your end-to-end infrastructure, entire real-time – near-real-time RIC, et cetera. So, as we build on the solutions towards a closed-loop automation, AI plays an important role, for example.

Lilian Veras

Awesome, thanks, guys. Question number two here. “Do we see new opportunities for new players to achieve RAN intelligence?”

Hassnaa Moustafa

Yes, I think that's a great point. And again, that's the whole Edge ecosystem if we return back to this point. Of course, we see – the point is the network itself started by moving from dedicated functions on dedicated hardware to software-defined network functions. That was the first step. And now like container-based network functions, enabling already new players to build the software for the RAN and for the 5G Core. We see new players building the software for the RAN, and for the core, and collaborating hand-in-hand with the telco to deploy it.

AI brings an additional type of new players to build the intelligent applications, the intelligent applications for the RAN, the xApp. Every now and then we hear in the ecosystem, we hear in the open community, we learn about new companies, or companies who were doing software development. ISVs now turning on to build applications for intelligent RAN, for intelligent functions in the network. Because you need AI expertise here, which are new expertise, plus network expertise.

So, it triggers, of course, new ecosystem players for building AI applications for the network besides the players that came new already building software-defined functions for the MEC. We're working with all these players. This collaboration between Red Hat and Intel is very interesting in that sense. The labs that Mike is speaking about where we're planning to work with different partners in the ecosystem, to show how all the players can come together, building RAN intelligence for IoT Edge, for MEC Edge, for different Edge services.

Lilian Veras

Awesome, thank you. Question number three. “If we compare with AI for services and then user applications, how do we see AI training versus AI inference taking place for RAN intelligence and network intelligence in general?”

Hassnaa Moustafa

Oh, that's an interesting analogy. When we think of AI inference versus training, if it's for an application like video analytics, or time series data analytics, usually training happens offline in the cloud. And after that, inference is deployed at the Edge. That's the usual model.

And network intelligence, RAN intelligence is changing this model a bit, because we can't wait for offline training in the cloud as there are a lot of dynamics happens, the network condition changes so frequently. So, training – the RAN Intelligent Controller is a very good example for training. We spoke today about near-RT RIC or near-real-time RIC responsible mainly for the inference. But the rest of the picture, Mike showed in one of the slides, we have also the non-real-time RIC or non-RT RIC. Most of the time, training happens in the non-RT RIC, and it feeds decisions, or updates needed for the inference models to the near-RT RIC.

So, here, it's a balanced way. Training takes place as part of the network infrastructure in the non-real-time RIC. So, it's not completely offline. It's not exact real-time to the millisecond, but not completely offline, enabling getting data from the network during the network operation, doing training, and reacting in real-time.

We hear a lot of federated AI or federated learning, which is emerging in research, and now coming to deployment, the Edge deployment itself, meaning between the RICs themselves. We spoke today about the RIC, but if we look at a scalable deployment, and we're speaking with Red Hat in this collaboration, how we can enable themselves to have a federated view, to learn the network conditions, and take appropriate decisions in time.

So, the model is shifting. That's true. It's not the traditional training model happening offline in the cloud, but coming to the Edge and the network infrastructure.

Lilian Veras

Thank you, Hassnaa. We have one last question. “So, the QoE assurance microservices gives a view of containers serving from a single dashboard. Is that the case?”

Michael Recchia

So, just to clarify – I'm not sure I understand the question – but just to clarify what we meant by QoE assurance. So, these are xApps. These are xApps that are microservices designed, so each xApp is a microservice basically. And so, the job of those xApps is to essentially do the inferencing on the RAN, so look at the different KPIs on the RAN and then make an inference about what's happening on the RAN, and then do some proactive handover, if you will, of the UE that are on the cell or on the bad cell where the anomalies are happening. So, that's what we meant by QoE assurance. It's really not per se giving you a view of container serving from a single dashboard. That's not the intent of that. So, I'm not sure I understand the question fully.

But just to clarify what we mean here is that we're maximizing the quality of experience of the user, of the UE, and that's the use case – at least the initial use case that we're running. And it does run as microservices, it really has nothing to do with the dashboard per se. So, I don't know if I'm interpreting the question right.

Lilian Veras

Thanks, Mike. That's all right. The members of our audience can also contact Red Hat using the links that are provided here in this slide. So, this was our last question.

I would like to thank all of you for sharing such great information. And we would like to thank our audience for being present here today, and ask you to please give our team a rating for the live recording so that we may continuously improve the quality of our webinars.

This concludes our webcast. Thanks again, and I'll see you next time.

Hassnaa Moustafa

Thank you. Thank you, Lilian. Thank you, everyone. Thank you.

Sunku Ranganath

Thank you.