FPGA and GPU support

Background

FPGA and GPUs are becoming common place to take away the load from cores. Some of the use cases for FPGA and GPU include

- Machine Learning and Deep Learning inference/training offloads
- Crypto and compression offloads
- Protocol (such as IPsec and PDCP) offloads

Though fixed function accelerators do provide better performance, multiple accelerators are required in a given compute node if there are workloads of different types.

Being programmable, acceleration functions can be dynamically changed in FPGA/GPUs based on the need and hence FPGAs and GPUs are considered to be generic accelerators.

Openstack and Kubernetes orchestrators are adding support for FPGAs and GPUs.

ONAP being a service orchestrator (end-to-end), makes placement decisions for VNFs/workloads and hence the awareness of FPGA/GPU is required in ONAP like any other HPA feature.

Since Openstack is in advanced stages (in Rocky release) of providing FPGA support (via Cyborg project), initial support in ONAP would support Openstack based cloud regions. K8S support will be added later.

FGPAs can be used in following ways:

1. Pre-programmed with acceleration functions
2. Orchestrator (Cyborg) programmed acceleration functions
    a. Supports dynamic programming - Based on workload being brought up, Openstack programs the acceleration function in the compute node where workload is about to be brought up.
3. Workload programmed
    a. In this case, workload programs the FPGA.

From ONAP perspective, there is not much difference between pre programmed and orchestrator programmed. These two are commonly known as AFaaS (Acceleration Function as a Service). In case of workload programmed, it is called FPGAaaS (FPGA as a Service).

Since workload programmed FPGA is yet to be matured in Industry, AFaaS is considered initially. Rest of design note assumes the support for AFaaS.

Note on Openstack FPGA support

One good thing that is happening in Rocky release is that all resources, whether it is CPU, disk, memory, PCIe SRIOV VF or even FPGA/GPU, are represented in uniform way. And hence the request for resources via flavor is also becomes uniform.There are no longer unique way of representing flavor attributes. Please see this design note here : https://specs.openstack.org/openstack/nova-specs/specs/queens/approved/granular-resource-requests.html

In summary: Any resource request in flavor always starts with property resource<N>:<resource name>:<count>. Any associated trait for the resource or traits by themselves can be represented as trait<N>:<trait name>=required.

In case of FPGA and GPU, there is additional generic property is getting added, whose syntax is function<N>:<Function type>=required. Note that this keyword is not interpreted by NOVA. It is only interpreted by Cyborg.

One example:

- Say that Openstack instance supports ARRIA10 FPGAs and functions such as IPSEC_ACCELERATION, PUBLIC_CRYPTO_ACCELERATION, TENSORFLOW_ACCELERATION, COMPRESSION_ACCELERATION.
- Say that compute nodes have 3 FPGA cards in each of them.
- Say that Openstack operator thinks that there would be
    - Some workload types that require IPSEC_ACCLERATION and COMPRESSION_ACCELERATION together.
    - Some workload types that require TENSORFLOW_ACCELERATION (2 of them) and COMPRESSION_ACCELERATION together
    - Some workload types that require IPSEC_ACCELERATION, PUBLIC_CRYPTO_ACCLERATION and COMPRESSION_ACCELERATION together

Openstack operator creates following flavors

Flavor 1:

- resource1:CUSTOM_ACCELERATOR_FPGA=1
- trait1:FPGA_INTEL_ARRIA10=required
- function1:IPSEC_ACCLERATION_AF=required
- resource2:CUSTOM_ACCELERATOR_FPGA=1
- trait2:FPGA_INTEL_ARRIA10=required
- function2:COMPRESSON_ACCLERATION_AF=required

Flavor 2:

- resource1:CUSTOM_ACCELERATOR_FPGA=2
- trait1:FPGA_INTEL_ARRIA10=required
- function1:TENSORFLOW_ACCLERATION_AF=required

- resource2:CUSTOM_ACCELERATOR_FPGA=1
- trait2:FPGA_INTEL_ARRIA10=required
- function2:COMPRESSON_ACCLERATION_AF=required

Flavor 3:

- resource1:CUSTOM_ACCELERATOR_FPGA=1
- trait1:FPGA_INTEL_ARRIA10=required
- function1:IPSEC_ACCLERATION_AF=required
- resource2:CUSTOM_ACCELERATOR_FPGA=1
- trait2:FPGA_INTEL_ARRIA10=required
- function2:PUBLIC_CRYPTO_ACCLERATION_AF=required
- resource3:CUSTOM_ACCELERATOR_FPGA=1
- trait3:FPGA_INTEL_ARRIA10=required
- function3:COMPRESSON_ACCLERATION_AF=required

Some points to note are:

- Grouping of various properties together is achieved by <N> in resource, trait and function keywords.
- Even though FPGA accelerators typically are PCIe cards, PCIe vendor ID/device ID and VFs are hidden. They don't need to be specified.
- Note that in above examples, only the FPGA accelerators are taken as an example. In reality, flavors will have other resources too.

ONAP design considerations

- Leverage HPA functionality added in Multi-Cloud, OOF and A&AI.
- Add new HPA feature called "FPGA-AFaaS-Acceleration"
- Have following HPA feature attributes:
  - fpga_device. As per above examples, this attributes takes value "FPGA_INTEL_ARRIA10".
  - fpga_AF_function: As per above examples, this attributes takes values such as IPSEC_ACCELERATION_AF, TENSORFLOW_ACCELERATION_AF, PUBLIC_CRYPTO_ACCELERATION_AF or COMPRESSION_ACCELERATION_AF
  - fpga_AF_count: This holds the number of AFs supported by the site for each workload

Changes in R4:

Component	Changes
OOF	■ No changes are expected in OOF. Our design goal for HPA is to ensure that there are no changes to the OOF for every new HPA feature.
A&AI	• No changes are expected

Multi-Cloud	• No changes in existing Openstack plugins
	• Introduce new Openstack plugin for Rocky
	  ○ Flavor discovery and flavor discovery in each flavor to follow new method introduced in Rocky release (resource, trait and function tuples)
	  ○ Support for following features
	  ■ Continue to support following features
	  • VCPUS
	  • Memory
	  • Disk
	  • NUMA
	  • Huge pages
	  • NI acceleration
	  • PCIe passthrough (Normal and SR-IOV)
	  • OVS-DPDK
	  • Core affinity and thread affinity policies
	  ■ Additional support
	  • FPGA
	  ■ Stretch
	  • GPU

Discovery by Multi-Cloud:

<Take above example and show how those FPGA feature is represented in A&AI>

HPA policies

<Show few examples on how HPA policies would look like>

TOSCA compute requirements

<Show few examples on how TOSCA compute requirements would look like>