Coarse-grained acceleration for Virtual Machines

2020-05-26T12:10:38Z

During the last decade (or maybe more), there has been a huge debate whether the ideal way to deploy applications lies in the traditional, monolithic way or via distributed microservices. Both methods have their advantages and disadvantages and we can see trade-offs taken both ways, regarding efficiency, simplicity, scalability, availability, performance etc.

In the era of Kubernetes, Containers, and MicroVMs, people talking about deploying a monolithic application are finger-pointed and called dinosaurs! On the other hand, the micro-service ecosystem has yet to show significant progress in more traditional execution paradigms, such as off-loading specific tasks to non-generic compute elements, like GPUs, FPGAs, or hardware acceleration engines in general.

Most platforms supporting distributed micro-services deployment rely on generic abstractions to provide the relevant hardware support. For instance, serverless platforms claim they support GPUs by simply adding the hardware component in the underlying infrastructure (eg. choosing an AWS GPU instance). This, however, introduces a semantic gap: users are forced to embed complex frameworks to drive the accelerator hardware in their micro-service application. Think of it like embedding the CUDA software stack in a serverless function. Crazy huh?

Moreover, the concept of microservices and distributed applications in general, refer to an alternate programming model; not to mapping software components of a monolithic design to individual functions. For example, off-loading compute-intensive tasks to hardware accelerators is, by definition, a long-running task; people consider the trade-off of copying input and output data to a device versus the compute capabilities of processing the data, and decide whether to off-load it to a gpu or just do the processing on the CPU itself. For this capability to be used on a microservices-based application, one has to careful consider the functional and data dependencies and come up with a design that keeps the application runtime at acceptable levels, while using the underlying resources efficiently.

Considering all the above, we came up with an idea of combining microservice-based application execution with hardware acceleration. We figured that all the application needs is to off-load a task; all the hardware needs is a task to run. Then let's combine those two, integrate them to a virtualized environment and design a middleware where, based on the hardware capabilities, the application is presented with a specific set of "acceleratable" coarse-grained functions.

The most indicative example of the above concept is when a task running on a VM wants to do image classification: (a) an image is presented to the application running on a VM; (b) the VM asks the underlying software layer to classify it based on a specific model (or specific rules); (c) the system off-loads this computation to a hardware accelerator and returns the result to the VM.

Now let us consider our current options: (i) passthrough the hardware accelerator to the VM running the task. This gives full processing power to the VM; however, it keeps the hardware resource idle when the VM is not using it – switching a device to multiple VMs using device passthrough is something time-consuming and is generally considered a bad idea. (ii) use API-remoting (e.g. rCUDA) to forward acceleration requests to the device via the network. This adds a significant amount of overhead since the control and data path crosses a network stack multiple times – let alone the CPU utilization involved in the end-to-end process.

We introduce a simple, yet powerful API for tasks running on VMs to use hardware acceleration functionality, abstracting away any detail on the hardware-specific software stack. We implement this using virtio supporting both QEMU VMs and Firecracker MicroVMs. We focus on two example use-cases: (1) off-loading crypto functions and (2) classifying images. We showcase the system's benefits on an NVIDIA Jetson nano board, enabling efficient and secure acceleration on Edge devices.