Pathways to Convergence – An Additional Scenario

Dennis Gannon

The frontier of computing is now, literally, at the edge of the network. The edge has long been the home of content distribution systems where content can be cached and access quickly, but now that the Internet-of-Things (IoT) is upon us, it is also increasingly important to do some of the computing at the edge. For example, if you have a thousand tiny sensors in a sensitive environment or farm and you need to control water from sprinklers or detect severe weather conditions, it is necessary to gather the data, do some analysis and signal an action. If the sensors are all sending WIFI messages they may be routable to the cloud for analysis, but a more common solution is to provide local computing that can do some event preprocessing and response while forwarding only summary data to the cloud or HPC system. That local computing is called the Edge, or if distributed as a system of edge servers, it is often called the Fog. These points are well documented in the report "Big data and extreme-scale computing: Pathways to Convergence-Toward a shaping strategy for a future software and data ecosystem for scientific inquiry" by Asch et. al. (Hereafter referred to as the Pathways Report.) In this note we discuss an additional paradigm of computing not covered in that report: hardware and software microservices.

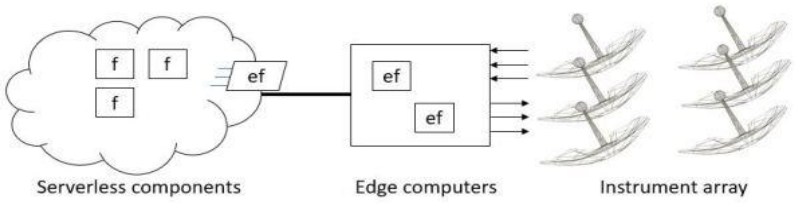
A microservice based application is based on decomposing the problem into several layers of independent software components that can be deployed and scaled independently. Suppose you are designing an application that must respond in near real-time to thousands of independent concurrent inputs. One layer of microservices can catch and clean the incoming data, which can then be forwarded to a second layer of analysis services that classify the input. A third layer can handle data logging, and another can deal with responses. The number of instances in each layer can be adjusted to match the input bandwidth requirements and scaled back when not needed. All the large commercial cloud providers base many, if not all, of their on-line services on this design. Examples include Netscape, Amazon.com, Twitter, Ebay, Uber, Google search, Bing and Azure's planet-scale CosmosDB. Applications running with thousands of instances are common. In addition to dynamic scaling, another advantage of this design paradigm is that services can be upgraded and replaced without taking the system down. (This is critical to DevOps design principles.)

Microservices are often packaged and deployed as containers running in massive-scale service fabrics like Kubernetes (open sourced by Google and now a standard). A related technology is Serverless computing in which the container instance is managed by a system that invokes the service as a function in response to an input signal. (Hardware microservices are small specialized devices that are deployed with the

servers to execute microservices. A great example of hardware microservices is the FPGA nodes in the Azure Project Brainwave.)

Migrating Microservices to the Edge.

If serverless functions are designed to respond to signals, that suggests that it should be possible to extend them to run in the edge servers rather than the cloud. AWS was the first to do this with a tool called GreenGrass that provides a special runtime system that allows us to push/migrate lambda functions or microservices from the data center to the edge. More recently Microsoft has introduced (and now open-sourced) Azure IoT Edge which is built on open container technologies. Using an instance of the open source Virtual Kubelet deployed on the edge devices, we can push our Kubernetes containers to run on the edge. You can think of a Kubelet as the part of Kubernetes that runs on a single node. This enables Kubernetes clusters to span across the cloud and edge as illustrated below.



Dynamically deploy edge functions from the cloud to edge or fog nodes as needed to optimize performance.

A major difference between this cloud-to-edge Kubernetes model and traditional HPC is that the former is optimized for long running systems while the latter is based on batch execution of tasks. IoT related activities require uninterrupted service. An ideal hybrid of cloud-edge and HPC would be allowing part of the HPC system to be partitioned off so that a system like Kubernetes can manage long-running containerized services (and interactive environments). Long-running workflow management services can adapt in real-time to events from the edge and submit big analysis singularity containers to the HPC side for execution.