Towards Network-Aware Resource Provisioning in Kubernetes for Fog Computing applications

Abdul Ahad Ayaz

Summary

In recent years, with the evolution of technology, IoT devices are increasing day by day. These IoT devices serves the mankind in different ways such as smart cites, smart transportation, water and waste management. With this increasing number of devices, the management and communication between these devices is one of the main issue. Fog computing addresses this issue by connecting heterogeneous IoT devices in a network and all the processing is done at the device level. Fog computing is the extension of cloud computing but decentralized.

Fog computing is responsible for providing resouces to IoT devices for processing. Traditionally these resources are allocated as VMs from different cloud infrastructure such as AWS, Google, OpenStack etc. to run the applications. VMs are considered resource greedy and require more computaional resource. Alternate is to use the Containers such as Docker[cite] which are light-weight,requires less resources and based on micro-service archietecture. Large applications are split into containers based on the main processes of the application. This increasing number of containers per application required the proper monitoring for health check and resource consumption. The most commonly used orchestrator for containers is Kubernetes[cite].

Kubernetes is an open-source platform for management,deployment and scaling of containers. Kubernetes archietecture is based on master slave model. Its consist of one master node and multiple worker node. Worker node provides the compute power. Master node communicates with the worker nodes using API, it is also responsible scheduling and deploying the containers across the cluster of worker nodes. Application consisting of multiple containers are deployed as a pod in Kubernetes. Each pod consist of only one IP address and all the underlying containers communicate using different ports. Different pods are isolated from each other and underlaying containers cannot communicate across pods. After receiving the pod configurations from user, master node schedule these pods based on the resources availability on different worker nodes.

When the new pod configurations are provided, pods are added to the waiting queue. Default Kubernetes scheduler monitors the waiting queue for pods and deploy the pod on specific worker node based on scheduling mechanism. scheduling is performed in two steps, first is the ”node filtering” and second is ”node priority calculation”. In first step worker node is selected by applying filter[paper8] such as ”PodFitsResources”, ”NoDiskConflict”, ”PodFitsHostPorts” etc. Sellected nodes are then prioritize in second

step by calculating ”LeastRequestPriority”, ”MostRequestedPriority”, ”CalculateAnti-AffinityPriority” etc. These scheduling is performed by considering resources such as CPU, memory, disk etc and network is not considered.

Considering one usecase of IoT such as smart cities, one of its service is to perform the anomaly detection as discussed in [paper22].

References