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 serve mankind in different ways such as smart cities, smart transportation, water, and waste management. With this increasing number of devices, the management and communication between these devices are one of the main issues. 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 resources to IoT devices for processing. Traditionally these resources are allocated as VMs from different cloud infrastructures such as AWS, Google, OpenStack, etc. to run the applications. VMs are considered resource greedy and require more computational resources. Alternate is to use the Containers such as Docker which are light-weight, requires less resources and based on micro-service architecture. 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.

Kubernetes is an open-source platform for management,deployment and scaling of containers. Kubernetes architecture is based on the 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. An 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 underlying containers cannot communicate across pods. After receiving the pod configurations from the user, master node schedules these pods based on the availability of resources 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 deploys the pod on a specific worker node based on a 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 filters [1] such as ”PodFitsRe-

sources", "NoDiskConflict", "PodFitsHostPorts" etc. Selected nodes are then prioritized in second step by calculating "LeastRequestPriority", "MostRequestedPriority", "CalculateAntiAffinityPriority" etc. These scheduling is performed by considering resources such as CPU, memory, disk etc and network is not considered.

Considering one use-case of IoT such as smart cities, where IoT applications collect the air quality data. Data is considered sensitive, making sure that no data is lost while it transmitted, making network configuration one of the main issues while deploying such applications. Default Kubernetes scheduler does not check for the latency and available bandwidth at a particular worker node. Considering this drawback, the author proposed a new network-based scheduler for pod deployment across worker nodes. This network scheduler is an extension of the default Kubernetes scheduler. This scheduler works in two steps, firstly calculates the Round Trip Time(RTT) to the target node mentioned in the pod configuration file and selects the node with minimum RTT. Second, it calculates the bandwidth of node and compares it with the bandwidth provided in the pod configuration based on best-fit selection. Results show that this network-based scheduler can improve the performance of Kubernetes by 80% in terms of network latency[9].

Related Papers

1. In [10], "Fogernetes"- fog computing platform is presented that manages and deploys fog applications.

2. In [4], "Firmament" - centralized scheduler. It schedules the tasks considering the network latency.

3. In [7], considering scheduling issues in 5G infrastructure, author propsed solution based on considering physical, operational and network parameters along with the software states.

4. In [3], dicussed about scheduling issues in fog computing, mentioned three scheduling policies that can improve the execution of application.

5. In [5], presented the Kubernetes scheduler that consider the application delay constraints.

6. In [8], presented the Dynamic Scheduling for Seamless Computing(DYSCO), a Kubernetes based scheduler.

7. In [6], presented the analysis on performance of Kubernetes deployment through Petri net-based performance model.

8. In [2], presented the an effective algorithm and architecture for resources provisioning in fog computing environment by employing virtualization technology.

References

[1] Production-Grade Container Orchestration - Kubernetes. URL https://kubernetes.io/.

[2] Swati Agarwal, Shashank Yadav, and Arun Kumar Yadav. An Efficient Architecture and Algorithm for Resource Provisioning in Fog Computing. 2016. http://j.mecs-press.net/ijieeb/ijieeb-v8-n1/IJIEEB-V8-N1-6.pdf.

[3] Luiz F. Bittencourt, Javier Diaz-Montes, Rajkumar Buyya, Omer F. Rana, and Manish Parashar. Mobility-Aware Application Scheduling in Fog Computing. 2017. https://ieeexplore.ieee.org/document/7912261.

[4] Ionel Gog, Malte Schwarzkopf, Adam Gleave, Robert N M Watson, and Steven Hand. *Firmament: Fast, Centralized Cluster Scheduling at Scale Firmament: fast, centralized cluster scheduling at scale*. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2016), 2016. https://www.usenix.org/conference/osdi16/technical-sessions/presentation/gog.

[5] David Haja, Mark Szalay, Balazs Sonkoly, Gergely Pongracz, and Laszlo Toka. Sharpening Kubernetes for the Edge. SIGCOMM 2019 - Proceedings of the 2019 ACM SIGCOMM Conference Posters and Demos, Part of SIGCOMM 2019, 2019. https://dl.acm.org/doi/10.1145/3342280.3342335.

[6] Víctor Medel, Rafael Tolosana-Calasanz, José Ángel Bañares, Unai Arronategui, and Omer F. Rana. Characterising resource management performance in Kubernetes. 2018. http://zaguan.unizar.es/record/75661/files/texto_completo.pdf.

[7] Benedek Kovács Michael Chima Ogbuachi, Anna Reale, Péter Suskovics. Context-Aware Kubernetes Scheduler for Edge-native Applications on 5G. 2020. https://jcomss.fesb.unist.hr/index.php/jcomss/article/view/1027.

[8] Ludwig Mittermeier, Florian Katenbrink, Andreas Seitz, Harald Muller, and Bernd Bruegge. Dynamic scheduling for seamless computing. Proceedings - 8th IEEE International Symposium on Cloud and Services Computing, SC2 2018, 2018. https://ieeexplore.ieee.org/document/8567371.

[9] Jose Santos, Tim Wauters, Bruno Volckaert, and Filip De Turck. Towards network-Aware resource provisioning in kubernetes for fog computing applications. IEEE Conference on Network Softwarization Unleashing the Power of Network Softwarization, NetSoft 2019, 2019. http://physics.nist.gov/Document/sp811.pdf.

[10] Cecil Wöbker, Andreas Seitz, Harald Mueller, and Bernd Bruegge. Fogernetes: Deployment and management of fog computing applications. IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS 2018, 2018. `https://ieeexplore.ieee.org/document/8406321`.