Chapter 10
Container Orchestration With Cost-Efficient Autoscaling in Cloud Computing Environments

Maria Rodriguez
University of Melbourne, Australia

Rajkumar Buyya
University of Melbourne, Australia

ABSTRACT

Containers are widely used by organizations to deploy diverse workloads such as web services, big data, and IoT applications. Container orchestration platforms are designed to manage the deployment of containerized applications in large-scale clusters. The majority of these platforms optimize the scheduling of containers on a fixed-sized cluster and are not enabled to autoscale the size of the cluster nor to consider features specific to public cloud environments. This chapter presents a resource management approach with three objectives: 1) optimize the initial placement of containers by efficiently scheduling them on existing resources, 2) autoscale the number of resources at runtime based on the cluster's workload, and 3) consolidate applications into fewer VMs at runtime. The framework was implemented as a Kubernetes plugin and its efficiency was evaluated on an Australian cloud infrastructure. The experiments demonstrate that a reduction of 58% in cost can be achieved by dynamically managing the cluster size and placement of applications.

INTRODUCTION

Containers, such as Docker (Docker, n.d.) and Linux Containers (LXC) (Bernstein, 2014), are standalone and self-contained units that package software and its dependencies together. Similar to Virtual Machines (VMs), containers are a virtualization technique that enable the resources of a single compute node to be shared between multiple users and applications. However, while VMs virtualize resources at the hardware level, containers do so at the operating system level. This makes them a lightweight

DOI: 10.4018/978-1-7998-2701-6.ch010

virtualization approach that enables application environment isolation, fast and flexible deployment, and fine-grained resource sharing.

Organizations are increasingly relying on containerization to deploy diverse workloads derived from modern-day applications such as web services, big data, and IoT applications. Container orchestration platforms, such as Kubernetes (Hightower et al., 2017), Docker Swarm (Naik, 2016) and Apache Mesos (Hindman et al., 2011), are responsible for the efficient orchestration of such applications in shared compute clusters. These platforms manage the lifecycle of containers as well as the usage of cluster resources and hence, one of their main goals is to (near) optimally place containerized applications on the available nodes. This scheduling problem is the focus of this chapter.

As applications are submitted for deployment, the orchestration system must schedule them as fast as possible on one of the available resources. This must be done while attempting to maximize the utilization of the cluster compute resources, as this is likely to reduce the operational cost of the organization. This placement should also be done while considering factors such as the capacity of the available machines, application performance and Quality of Service (QoS) requirements, fault-tolerance, and energy consumption among others. Although existing container orchestration platforms address these issues to an extent, further research is required in order to better optimize the use of resources under different circumstances and for different application requirements.

In cloud environments, containers and VMs can be used together to provide users a great deal of flexibility in deploying, structuring, and managing applications. In such cases, not only should the number of containers scale to meet the requirements of applications, but the number of available compute resources should also adjust to adequately host the required containers. At any given point in time, a cloud container orchestration system should avoid underutilizing VMs as a cost and energy controlling mechanism. It should also be capable of dynamically adding worker VMs to the cluster in order to avoid a degradation in the applications' performance due to resource overutilization. Therefore, autoscaling the number of VMs is essential to successfully meeting the performance goals of containerized applications deployed on public clouds and to reduce the operational cost of leasing the required infrastructure. This increases the complexity of the container placement and scheduling problem mentioned above.

Existing container orchestration frameworks provide bin-packing algorithms to schedule containers on a fixed-sized cluster but are not enabled to autoscale the size of the cluster. Instead, this decision is left to the user or to external frameworks at the platform level. An example of such a scenario is using Kubernetes for the placement of containers and Amazon's autoscaling mechanism to manage the cluster nodes. This approach may not only be impractical but also inefficient as external entities have limited information regarding the container workload.

Another optimization facet not considered by leading orchestration systems is related to rescheduling; in particular, rescheduling to reduce resource fragmentation and to reduce the size of the cluster. Regardless of how good the initial placement of these tasks is, the utilization of resources will degrade over time as the workload changes. Rescheduling applications that tolerate a component being shut down and restarted will enable the orchestration system to consolidate and rearrange tasks so that more applications can be deployed on the same number of nodes or some nodes can be shutdown to reduce cost or save energy. Similarly, if more nodes are added to the cluster, being able to reschedule some of the deployed applications on the new nodes may be beneficial in the long term.

As a result, the work on this chapter argues that a cloud-centric container orchestration framework capable of making all of the resource management decisions, including autoscaling and rescheduling, is essential in successfully optimizing the use of resources in cloud environments. The main contribution