The Curious Case of Container Orchestration and Scheduling in GPU-based Datacenters

Extended Abstract

Prashanth Thinakaran[1], Jashwant Raj[1], Bikash Sharma[2], Mahmut T. Kandemir[1] and Chita R. Das[1]
[1]Pennsylvania State University, [2]Facebook Inc

ABSTRACT

Modern data centers are increasingly being provisioned with compute accelerators such as GPUs, FPGAs and ASIC's to catch up with the workload performance demands and reduce the total cost of ownership (TCO). By 2021, traffic within hyperscale datacenters is expected to quadruple with 94% of workloads moving to cloud-based datacenters according to Cisco's global cloud index. A majority of these workloads include data mining, image processing, speech recognition and gaming which uses GPUs for high throughput computing. This trend is evident as public cloud operators like Amazon and Microsoft have started to offer GPU-based infrastructure services in the recent times.

The GPU-bound applications in general, can either be batch or latency-sensitive. Typically the latency-critical applications subscribe to datacenter resources in the form of queries (e.g. inference requests from a DNN model). For example, a wearable health monitoring device aggregates several sensor data through a mobile application. In case of a data anomaly, inference services can be triggered from the mobile device to the cloud, requesting for a deep neural network (DNN) model that fits the symptom. Such inference requests which are GPU bound impose strict Service Level Agreements (SLAs) that is typically set around 150 to 500ms. In contrast to the regular datacenter batch workloads, these user-facing applications are typically hosted as services that occur and scale in short bursts. On the other hand, batch applications are HPC based compute-bound workloads which are throughput oriented. In a typical datacenter, these both applications might co-exist on the same device depends on the orchestration and scheduling policy. With the expected increase in such workloads, this GPU resource management problem is expected to exacerbate. Hence, GPUs/accelerators are on the critical path to ensure the performance and meet the end-to-end latency demands of such queries.

State-of-the-art resource orchestrators are agnostic of GPUs and their resource utilization footprints, and thus not equipped to dynamically orchestrate these accelerator-bound containers. On the other hand, job schedulers at the datacenter are heavily optimized and tuned for CPU-based systems. Kubernetes and Mesos by default does uniform task scheduling which statically assigns the GPU resources to the applications. The scheduled tasks access the GPUs via PCIe pass-through which gives the application complete access

SoCC '18, October 11–13, 2018, Carlsbad, CA, USA

ACM ISBN 978-1-4503-6011-1/18/10.
https://doi.org/10.1145/3267809.3275466

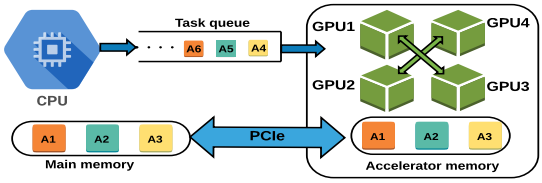

Figure 1: Application queue on a node with multiple GPUs.

to the GPU as seen in Figure 1. Hence the resource utilization of the GPU is based on the parallelism of the application which is scheduled to run on it. In case of CPUs, Kubernetes has support for dynamic orchestration with the features such as node affinity, pod affinity, and pod preemption. However, these features cannot be extended for GPUs. This is because, it neither has the support for pod preemption nor the ability to query the real-time GPU metrics such as memory, symmetric multiprocessor (SM) utilization, PCIe bandwidth, etc. Moreover, the containers often overstate their GPU resource requirements such as memory, and this leads to severe resource underutilization which leads to multiple QoS violations because of queuing delays. We identify that by employing CPU-based scheduling policies for GPU-bound workloads would fail to yield high accelerator utilization and lead to poor performance per watt per query. Motivated by this, we propose a GPU-aware resource orchestration layer which enables the resource scheduler to take advantage of the GPUs by knowing their real-time utilization.

We further discuss the ideal scheduler properties for a GPU rich datacenter and list the challenges in developing such a production-grade GPU-based datacenter scheduler. Therefore we modify the well-known Google's Kubernetes datacenter-level resource orchestrator by making it GPU-aware by exposing GPU driver APIs. Based on our observations from Alibaba's cluster traces and real hardware GPU cluster experiments, we build *Knots*, a GPU-aware resource orchestration layer and integrate it with *Kubernetes* container orchestrator. In addition, we also evaluate three GPU-based scheduling schemes to schedule datacenter representative GPU workload mixes through *Kube-Knots*. Evaluations on a ten node GPU cluster demonstrate that *Knots* together with our proposed GPU-aware scheduling scheme improves the cluster-wide GPU utilization while significantly reducing the cluster-wide power consumption across three different workload mixes when compared against Kubernetes's default uniform scheduler.

ACKNOWLEDGMENTS

This research was generously supported by several NSF grants 1320478, 1409095, 1629129, 1439021, 1629915, 1439057, 1626251, 1526750 and NSF Chameleon Cloud project CH-819640 for the GPU cluster infrastructure.