Improving Quality of Service (QoS) of Multi-Tenant GPU Clouds Enabled by Passthrough

Youssef Elmougy and Jianchen Shan Ph.D.
Fred DeMatteis School of Engineering and Applied Science, Hofstra University

ABSTRACT

GPU-accelerated cloud has become the mainstream with explosive demand from applications such as deep learning. Many studies have been conducted to improve GPU virtualization technology to increase efficiency while sharing GPU among multiple tenants. Little attention has been paid to the performance of GPU instance using passthrough technique. The Quality of Service (QoS) of GPU in such instance is assumed to be guaranteed since the GPU is dedicatedly attached to the instance without being shared. However, through some motivating experiments, we show that the passthrough-attached GPU could be severely underutilized in the multi-tenant server, and the GPU utilization can be degraded up to 30x when compared to the GPU instance running in the dedicated server. The key observation is that the multi-tenant server GPU instance would still share the multicore CPU with other instances for higher system throughput and lower cost. A typical GPU application would prepare the data and the task (i.e., the GPU kernel) on CPU and transfer them to the GPU to launch the kernel. However, the interference from other instances on the CPU side could delay the dispatch and response of the GPU kernel, and cause low GPU QoS. To mitigate this problem, application characteristics would be analyzed to identify the performance bottleneck, based on which the system level solution, such as an amendment in the priority of processes, would be proposed.

BACKGROUND

GPGPU (General Purpose GPU) greatly accelerates large scale data processing and parallel computing. Thus, most major cloud providers have introduced the GPU cloud by provisioning the GPU instances.

Cloud Provider	vGPUs	Available vCPUs
Microsoft Azure [1]	1, 2, 4	6, 12, 24
Amazon Web Services (AWS) [2]	1, 4, 8	8, 32, 64
Google Cloud [3]	1, 2, 4	16, 32, 64
IBM Cloud [4]	1, 2	16, 28, 36
Oracle Cloud [5]	1, 2, 4, 8	12, 24, 28, 52

GPU virtualization, as the key enabling technology, allows GPU instances to access either the shared or dedicated GPU devices. Several main techniques include:

- PCI Passthrough (DirectPath I/O)
- NVIDIA GRID vGPU
- intel KVMGT
- AMD MxGPU

PCI Passthrough [6] is the most widely used technique which supports high performance cloud infrastructure. It is the method by which a Virtual Machine (VM) gains direct access to the dedicated GPU, permitting the VM to fully benefit from the GPU's processing capability.

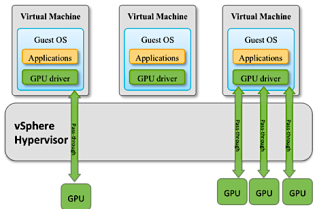
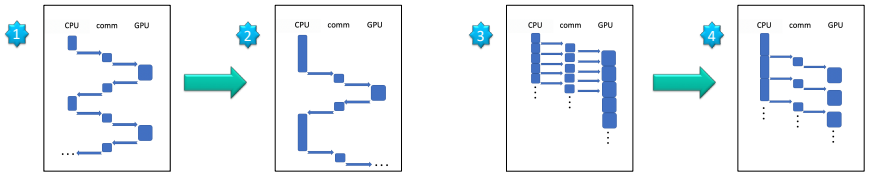


Fig. PCI Passthrough System Architecture

PROBLEM STATEMENT

The dividing and issuing of tasks between the CPU and the GPU is an essential asset in program execution. This involves 3 major phases: 1) CPU execution (which includes setting up variables, memory allocation, execution of code snippets, sending of kernels to GPU), 2) CPU-GPU communication (which includes copying data between CPU->GPU and GPU->CPU), 3) GPU execution (which includes extracting instructions, executing and computing data). The following flowcharts demonstrate the different scenarios as to which the 3 phases may be executed:



In the first scenario, the CPU and GPU are processing synchronously and no additional kernels are sent until the data is fully processed and sent back to the CPU. In the third scenario, the CPU and GPU are processing asynchronously. Specifically, it shows that the CPU sends kernels faster than the GPU can process. This forms a cluster of kernels hence keeping the GPU fully utilized. The problem is when the CPU execution is slowed down by the resource sharing, kernel dispatching and responding could be severely delayed. This would convert the first scenario to the second scenario and convert the third scenario to the fourth scenario. As shown in the second and fourth scenarios, if the CPU phase is delayed due to sharing resources with other VMs, the GPU utilization would be reduced as a consequence, and have an overall negative effect on the performance of the workload being processed.

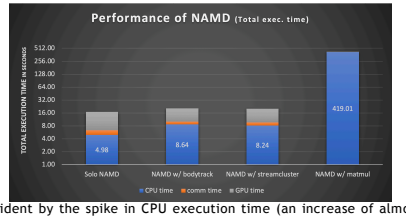
MOTIVATING EXPERIMENT

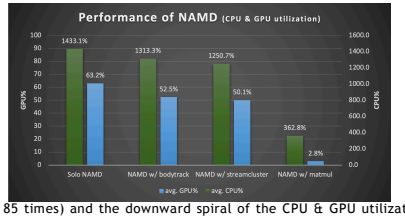
To show the GPU underutilization problem, experiments were conducted on the JUPITER server, which contains 4 Intel Xeon Gold 6138 CPUs with 80 cores@2.00GHz and 2 NVIDIA Corporation Tesla P100 PCIe 16GB GPUs. Two Virtual Machines (VMs) are setup as a representation of the multi-tenant GPU cloud. The first VM is a 16-vCPU VM with a dedicated GPU attached by PCI Passthrough to run the CPU- & GPU-intensive workload. The second VM, as the corunner VM, is a 16-vCPU VM to run only the CPU-intensive workload without any GPU attached. The two VMs are pinning on the same set of 16 cores to share the CPU resource. The workloads/benchmarks that were exploited in this experiment were chosen due to their extensive use in countless research projects and their focus on different fields of study. The following are the two categories of benchmarks chosen and their fields of study:

Both CPU & GPU Intensive: {CAT1}
- NAMD [Scientific Simulation]
- Gromacs [Biochemical Molecular Dynamics]
- DQN [Artificial Intelligence]

Uniquely CPU Intensive (used in corunner VM): {CAT2}
- Streamcluster [Data Mining]
- matmul [Matrix Multiplication]
- Bodytrack [Computer Vision]

Each benchmark in CAT1 was evaluated in the following two scenarios: 1) when the benchmark was running in the first VM without corunner VM running, 2) when the benchmark was running in the first VM with the corunner VM executing each benchmark in CAT2. The following shows the results of the NAMD benchmark when it was executed in the two scenarios:





It is evident by the spike in CPU execution time (an increase of almost 85 times) and the downward spiral of the CPU & GPU utilization (a decrease of almost 30 times) that the performance of the benchmark was degraded as compared to when it was run solo. This puts light and proves that there exists a problem that causes heavy turnover and underutilization of resources on cloud platforms that are potentially running a colossal number of instances to consumers.

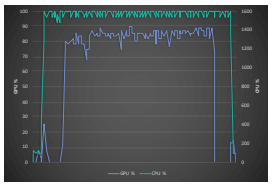


Fig. CPU & GPU % during solo NAMD

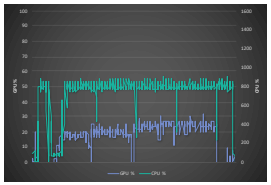
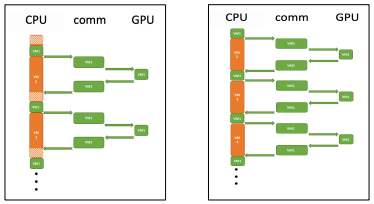


Fig. CPU & GPU % during NAMD w/ matmul

PROPOSED SOLUTION (work in progress)

We use an example here to explain the key idea of the proposed solution. In the figures below VM 1 is running the GPU workload and VM 2 (i.e., the corunner) is running the CPU workload. The two VMs are sharing the CPU resource. As the left subfigure shows, whenever the VM1 tries to send data/kernel or respond to the result from the GPU, the CPU execution is delayed by VM2's execution, which is indicated by the shadow box. This is because by default the work conserving scheduling is used. When VM1 is idle on CPU, VM2 would take advantage of the CPU resource to reduce the waste. However, this would slow down the speed of VM sending and responding to the data/kernel and reduce GPU utilization because of the delay. We proposed to temporarily boost the priority of VM1 once the system detects that VM1 is handling the GPU data/kernel as shown in the right subfigure while maintaining the fairness. In the best case, VM1 would be running like using CPU delicately without causing the GPU underutilization problem. The proposed idea can be applied for both synchronized and asynchronized executions.



FUTURE WORK

The following ideas and experiments could be done:

- deeper analysis of already chosen benchmarks and the acquiring of more benchmarks in common fields of study
- more studying of CUDA programming to implement application-level modifications to benchmark program code
- producing and proposing a resilient system-level solution to lessen or eliminate the effects of the problem

REFERENCES

[1] Docs.microsoft.com. (2019). *Azure Windows VM sizes - GPU*. (https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sizes-gpu)

[2] Amazon Web Services, Inc. (2019). *Amazon EC2 Instance Types - Amazon Web Services*. (https://aws.amazon.com/ec2/instance-types)

[3] Google Cloud. (2019). *GPUs on Compute Engine*. (https://cloud.google.com/compute/docs/gpus/#gpu_comparison_chart)

[4] Ibm.com. (2019). *GPUs for Cloud Servers*. (:https://www.ibm.com/cloud)

[5] Cloud.oracle.com. (2019). *Accelerated GPU Cloud Computing* (https://cloud.oracle.com/compute/gpu/features)

[6] IBM Developer. (2019). *Linux virtualization and PCI passthrough*. (https://developer.ibm.com/tutorials/l-pci-passthrough)