Analyzing improvement methods in GPU virtualization towards the future

Christina Tang
Electrical & Computer Engineering, University of Washington

1 Introduction

The rise of machine learning, graphic-intensive software, and high-performance computing systems indicate an increased need for GPU usage. However, acquisition costs, overhead, and availability are constraints for companies to access more GPU processing power. GPU demand makes them costlier, computing facilities need space, and GPUs have high power consumption even when they are not in use. Previously, virtualization did not include GPUs, but CPU-based virtualization does not meet the increasing demand for computational power.

This has introduced cloud computing systems for remote GPU acceleration and the offering GPGPU services. Virtual GPU acceleration is now performing just as well as physical GPUs. New proposals and technologies in GPU virtualization methods aims to further improve this through improving resource utilization and efficiency to decrease power consumption, increase scalability, and provide further flexibility for consumers.

In addition, the growing field of Internet of Things, or IoT, benefits from remote GPGPU services. IoT devices are all connected to the cloud, so GPGPU services impact device management and data processing. With more IoT devices coming into existence, the ever-increasing demand for cloud computing power becomes very apparent.

This paper discusses approaches to this problem by discussing and comparing techniques in virtualization and containerization, and what could be the direction of these methods in the future.

2 Virtualization Methods

Remote GPU virtualization for GPGPU aims to solve increasing acquisition costs, space, and energy with GPU sharing in clusters. This method can effectively utilize all GPUs and scale to support more users. Cluster throughput, or jobs completed per time unit, has been shown to double with this technique. In addition, total energy consumption was reduced up to 40%.

There has also been a recent shift towards containerization. This method reduces virtual runtime, size, flexibility. It also integrates tasks like application packaging and service orchestration.

2.1 NVIDIA vGPU

NVIDIA's virtual GPU, or vGPU, software remotely accelerates desktop applications by splitting a physical GPU on a server into multiple virtual GPUs. This allows a GPU to be shared across many users. Splitting a GPU amongst users allows providers to allocate resources efficiently. It also provides flexibility to provide how much computation a user will need. In the past, virtual GPUs had a one-to-one relationship with the user; vGPU allows a one-to-many relationship between a GPU and its users.

Its software provides users with the experience of a physical GPU through each virtual machine. Workplace software from video conferencing to computer-aided design benefit from virtual GPUs without needing to purchase and distribute physical GPUs to individual employees. Many companies and institutions from industries in finance, satellite imagery, automotive, legal, education, and more, have utilized NVIDIA's virtual GPU technology to achieve higher performance, lower latency, and greater productivity.

2.2 rCUDA

rCUDA, or remote CUDA, is a framework enables GPU acceleration remotely on CUDA-enabled GPUs. CUDA is a programming model and parallel computing platform that helps developers use a GPU with better performance. In high performance computing clusters, each node typically has its own GPU, but having a GPU in each node in a cluster significantly increases power consumption. One extra node can consume 50% more energy, and a high-end GPU will consume even more. This is the motivation for decreasing the number of GPUs in clusters for significant energy savings. rCUDA reduces the number of accelerators in the cluster, which improves energy consumption, acquisition costs, maintenance, and physical space.

rCUDA applies itself as middleware to manage remote GPU accelerators. The client and server middleware of rCUDA replace functions from the CUDA library to reduce the number of accelerators in a cluster. It intercepts calls by making wrapper libraries around the CUDA runtime API on the client side, and having server-side middleware configured as a daemon that runs on the nodes with GPGPU acceleration services.

rCUDA decreases execution time even with negligible performance reduction.
In vCUDA, when the OS driver for the GPU is not being used, the initialization time takes longer.

vCUDA and rCUDA may be weak methods, because virtualization inherently degrades performance. In addition, they are all still subsets of the CUDA API, and the complete CUDA API documentation is not publicly available. This is a problem for proposals outside NVIDIA itself, because the CUDA compiler inserts hidden, undocumented functions and makes calls to them. rCUDA required rewriting functions to avoid the hidden calls. Every new version of CUDA requires all virtualization frameworks to reconfigure.

2.3 NVIDIA Docker

The downside to VMs is that there is an entire operating system for each virtual GPU. While the hypervisor-based virtual machines can run multiple applications, containerization virtualizes one application for each container. This means that containers are simpler to manage because applications do not make calls to host operating system kernels.

Docker itself is used for encapsulating application dependencies for deployment. Docker containers are reproducible and reliable, but also fast because containers initialize much faster without virtual machine overhead. It does not natively support NVIDIA GPUs, so NVIDIA Docker was created to do so. NVIDIA Docker provides GPU acceleration for applications, while simultaneously improving development, testing, and production workflow.

Microservices also benefit from NVIDIA Docker by integrating it with a serverless computing framework to high-performance microservices. CPU-based serverless computing frameworks already exist, so a GPU-supported framework can help developers deploy high-performance microservices with a negligible decrease in performance.

However, there are minor drawbacks to NVIDIA Docker. According to the NVIDIA Docker repository, it also does not support platforms macOS, windows, or ARM64 driver stacks. CUDA multi process service or GPU-accelerated X servers are also not yet supported. For developers already using the virtual machines, transitioning or migrating to containers may require extra time and effort because of the lack of complete functionality on these specific contraints.

2.4 ConVGPU

NVIDIA Docker containerization is not the best solution either, because is susceptible to deadlock. NVIDIA Docker has resource sharing and concurrent access, so multiple containers can access the same GPU memory and cause deadlock.

ConVGPU exists to solve this problem by building on top of NVIDIA Docker to manage GPU memory between containers. It isolates GPU memory, so each container does not use all the memory of an entire GPU. This container-based proposal with NVIDIA Docker proves it does not significantly decrease performance.

Another significant benefit is that ConVGPU is fully compatible with the CUDA API. This has a large benefit against middleware proposals like rCUDA, where its drawback was the lack of transparency with CUDA API.

3 Comparison

Containerization works well because of decreased hardware costs, reliability, scalability, and spatial isolation. Dockers were made for continuous integration and portability, which significantly improves deployment workflow. They are faster because containers can start up in a

few seconds, while a virtual machine can take minutes. Above all, containers have shown that they have a significant performance advantage.
Transitioning from virtualization to containerization reduces overhead costs and increases the performance for all common applications. However, specific uses like data clustering based on the k-means method is faster in a virtual machine. Not every application is benefitted using containers, so companies need to consider tradeoffs. Hybrid virtualization also creates other tradeoffs, like security in exchange for performance and real-time applications.

Overall, it may be worth the time and effort for most developers to switch to a container-based method if they can, and if their functional needs are supported. It may also be easier for developers to choose just one method across the board. However, it also depends on the ways containers are configured.

4. Considerations

Many of the proposed technologies and methods so far have boasted large improvements in energy and efficiency to virtualized GPUs without much degradation in time or performance. Looking towards the future, both containerization and VM-based GPUs methods need to consider other factors like security, new GPU generations, and how to price services from the increased flexibility.

4.1 Security

GPU virtualization is prone to confidentiality issues. The driver, operating system, hypervisor, and physical GPU have no memory cleanup for security. A bad actor can start a virtual machine right after the victim uses a virtual machine on the same GPU. Since there is no proper memory cleaning, the isolation of virtualization is bypassed and data leakage can happen. The recommendation is for developers to handle GPU memory just like main memory, and memory security can be implemented on several layers.

Although Docker is very popular as a complete packaging and software delivery tool, it has vulnerabilities. Insecure configuration and weak access control are the critical areas, but the issues can also be mitigated on several layers by limiting Docker to remove host dependence through abstraction. Cloud administrators and software architects suggest running containers on top of a virtual machine. However, NVIDIA Docker security issues have not been thoroughly studied, and could have a different implementation.

Overall, security is something that all developers need to keep in mind. Increasing code in production only creates a larger surface for malicious attacks.

4.2 New generations of GPUs

On a study of remote GPU virtualization performance over three generations of GPUs, the results show that the underlying GPU hardware has a big impact on performance. For each

generation, the virtualization middleware needs to change its design. This is another argument for containerization against virtualization. The study concludes that every new generation requires another adaptation.

4.3 Pricing

With the ability of giving more users remote GPU computing power, service providers are shifting focus towards GPU-accelerated services. There are now different ways to provide remote GPU services and greater flexibility for clusters in high-performance computing and different needs in microservices, IoT, and machine learning. Researchers suggest implementing a pricing strategy as a two-stage leader-follower (Stackelberg) game to analyze the equilibrium. The analysis will result in a better payoff for both the cloud provider and its users. However, this is just one study, and better pricing strategies may come about when analyzing GPU-accelerated cloud computing in the real world.

Given that many industries are turning towards using remote GPU virtualization, the economy is changing, and providers need to consider how to elastically price their services to stay competitive. The goal is to beat the acquisition cost of a company buying GPUs for their own use.

4 Conclusion

Currently, there is no one-size-fits all method for how virtual GPU acceleration services should be done. Concurrent research and development in this area is a race to find methods and solutions for secure, flexible, and energy efficient remote GPU virtualization.

Container architectures can be redesigned by reducing creation time for faster and safer function creation and deployment. More research needs to be done on charging policies for the cloud providers that provide the environments of virtual GPUs to users. ConVGPU can be improved by adopting a clustering system like Docker Swarm and its researchers claim extending it in a multiple GPU will achieve greater performance with an appropriate algorithm.

Although containerization is generally better, virtualization is much better in specific use cases. In addition, hybrid virtualization improves other use cases even more. The best methods are dependent on the providers, users, and the specific services. Further into the future, the methods will be ever-evolving with new security vulnerabilities and new hardware.

Improvements in distributed computing infrastructures are a must because of its broad impact. Every industry, even outside of technology, is demanding more computation power. With more use cases than ever, services may be able to effectively customize its GPU virtualization specific to each application.

References

[1] Silla, F., Prades, J., Iserte, S. and Reano, C. (2016). Remote GPU Virtualization: Is It Useful?. *2016 2nd IEEE International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*.

[2] Duato, J., Pena, A., Silla, F., Mayo, R. and Quintana-Orti, E. (2010). rCUDA: Reducing the number of GPU-based accelerators in high performance clusters. *2010 International Conference on High Performance Computing & Simulation*

[3] Kang, D., Jun, T., Kim, D., Kim, J. and Kim, D. (2017). ConVGPU: GPU Management Middleware in Container Based Virtualized Environment. *2017 IEEE International Conference on Cluster Computing (CLUSTER)*.

[4] Kim, J., Jun, T., Kang, D., Kim, D. and Kim, D. (2018). GPU Enabled Serverless Computing Framework. *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*.

[5] Radchenko, G., Alaasam, A. and Tchernykh, A. (2019). Comparative Analysis of Virtualization Methods in Big Data Processing. *Supercomputing Frontiers and Innovations*, 6(1).

[6] Maurice, C., Neumann, C., Heen, O. and Francillon, A. (2014). Confidentiality Issues on a GPU in a Virtualized Environment. *Financial Cryptography and Data Security*, pp.119-135.

[7] Combe, T., Martin, A. and Di Pietro, R. (2016). To Docker or Not to Docker: A Security Perspective. *IEEE Cloud Computing*, 3(5), pp.54-62.

[8] Reano, C. and Silla, F. (2017). A Comparative Performance Analysis of Remote GPU Virtualization over Three Generations of GPUs. *2017 46th International Conference on Parallel Processing Workshops (ICPPW)*.

[9] Li, H., Ota, K., Dong, M., Vasilakos, A. and Nagano, K. (2017). Multimedia Processing Pricing Strategy in GPU-accelerated Cloud Computing. *IEEE Transactions on Cloud Computing*, pp.1-1.