HPC Supported Mission-Critical Cloud Architecture

Tae Joon Jun, Myong Hwan Yoo, Daeyoung Kim
School of Computing, KAIST
Daejeon, Korea
{taejoon89, yoomhwan, kimd}@kaist.ac.kr

Kyu Tae Cho, Seung Young Lee
LIG Nex1 Corporation
Yongin, Korea
{kyutae.cho, seungyoung.lee}@lignex1.com

Kyuoke Yeun
Agency of Defense Development
Daejeon, Korea
koyeun@add.re.kr

ABSTRACT

Tactical Operations Center (TOC) system in military field is an advanced computer system composed of multiple servers and desktops to interlock internal/external weapon systems processing mission-critical applications in combat situation. However, the current TOC system has several limitations such as difficulty of integrating tactical weapon systems including missile launch system and radar system into the single TOC system due to the heterogeneity of HW and SW between systems, and an inefficient computing resource management for the weapon systems.

In this paper, we proposed a novel HPC supported mission-critical Cloud architecture as TOC for Surface-to-Air-Missile (SAM) system with OpenStack Cloud OS, Data Distribution Service (DDS), and GPU virtualization techniques. With this approach, our system provides elastic resource management over the weapon systems with virtual machines, integration of heterogeneous systems with different kinds of guest OS, real-time, reliable, and high-speed communication between the virtual machines and virtualized GPU resource over the virtual machines. Evaluation of our TOC system includes DDS performance measurement over 10Gbps Ethernet and QDR InfiniBand networks on the virtualized environment with OpenStack Cloud OS, and GPU virtualization performance evaluation with two different methods, PCI pass-through and remote-API. With the evaluation results, we conclude that our system provides reasonable performance in the combat situation compared to the previous TOC system while additionally supports scalable and elastic use of computing resource through the virtual machines.

CCS Concepts

•Networks → Cloud computing; •Applied computing → Military; •Computer systems organization → Heterogeneous (hybrid) systems; Dependable and fault-tolerant systems and networks;

ICPE'17, April 22-26, 2017, L'Aquila, Italy

DOI: http://dx.doi.org/10.1145/3030207.3044531

Keywords

Cloud Computing; Tactical Operations Center; Data Distribution Service; GPGPU;

1. INTRODUCTION

In the military field, multiple internal/external weapon systems such as radar system, early warning system, missile launch system, and satellite system are required for the SAM system in the combat situation. The TOC for SAM system is a composite computer system which controls above weapon systems for the SAM in the real combat situation. For example, the TOC for SAM system analyzes data from the radar tracking system while the missile launch system targets enemy forces in the combat situation. Fig. 1 shows an overview of the general TOC for SAM system.

However, the current TOC for SAM system has several limitations. Firstly, internal/external weapon systems are heterogeneous, mixing several operating systems, hardware requirements, programming languages, and development environments. Supporting an integrated control system for these heterogeneous systems with traditional server-client model is an inefficient approach and almost impossible to implement in real world. Secondly, the number of weapon systems used in the combat situation is different from time to time. For example, when the early warning system is in active state with the satellite and the radar systems, other

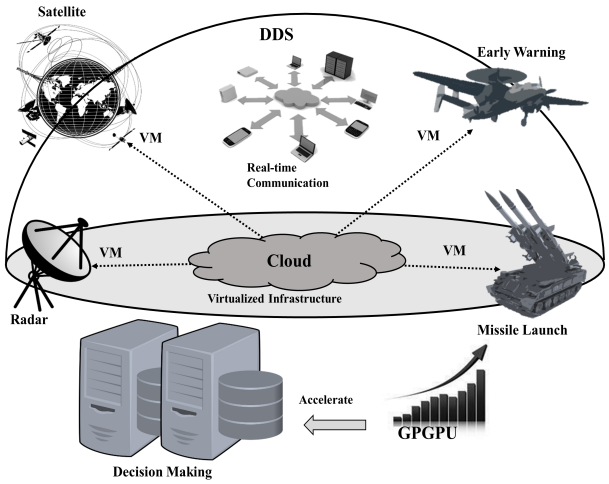


Figure 1: Overview of the TOC System

weapon systems such as air defense system and guided missile system may not be turned on. Providing computing resources to every weapon system is an extremely inefficient approach from the resource management view point. Finally, making decisions to control in the combat situation should be real-time and accurate. Communication between the weapon systems related to the SAM should be high-speed. In addition, analyzing combat data from the weapon systems should be accurate too.

In this paper, we proposed a novel HPC supported mission-critical Cloud architecture as TOC for SAM system with OpenStack Cloud OS [16], Data Distribution Service (DDS), and GPU virtualization techniques. To overcome the heterogeneous system problem, we used OpenStack open source cloud OS to provide virtualized infrastructure to the system. Every weapon system is loaded to the virtual machines created by OpenStack Cloud OS with different guest OS such as Windows and Linux. Cloud computing also enables the elastic use of computing resource for the weapon systems which overcomes the second problem stated above. DDS with high speed network, such as 10Gbps Ethernet and QDR InfiniBand, assures real-time communication between the weapon systems in the TOC system with the virtual machines. GPUs are used to accelerate algorithm that supports fast and accurate data analysis for the decision making. For example, Kalman filter algorithm for the radar tracking system can be written in GPGPU programming languages such as CUDA and OpenCL. GPU virtualization techniques including PCI pass-through and remote-API are used to provide virtualized GPU resources to the virtual machines. With DDS and GPU virtualization, our approach solves the third limitation stated above.

Evaluation of our TOC for SAM system processed in three different ways. First, we evaluated the DDS performance over two high speed networks, 10Gbps Ethernet and QDR Infiniband, in both non-virtualized and virtualized environments. Then, we experimented the DDS on the multiple virtual machines to check scalability of the DDS in the virtualized TOC system. Finally, we measured the performance of the real world GPU benchmark programs with two common GPU virtualization techniques, PCI pass-through and remote-API, on the virtual machines. From the evaluation, latency of the DDS over 10Gbps Ethernet with 1ms stable message interval shows average 5% faster than that of the QDR Infiniband. Between non-virtualized and virtualized environments, latency result with 1ms message interval time in the non-virtualized environment shows average 65% faster than that of the virtualized one. In case of 0.1ms message interval, we checked that the blocked messages at the publisher can significantly decreases the overall performance of the DDS in the TOC system. In addition, we also measured throughput, and standard error for the DDS performance evaluation. With the DDS performance results, we concluded that using DDS with the message period higher than 1ms can provide reliable and reasonable performance and scalability over the TOC system. In case of running GPU benchmark programs on the virtual machines with PCI pass-through and remote-API, execution time of the programs shows various results that are related to the characteristic of benchmark programs such as high memory copy proportion.

The paper is organized as follows. Section 2 provides background information and Section 3 covers the detail architecture of the system. Section 4 includes evaluation and experimental results of the system. Previous works related to our approach is provided in the Section 5. Conclusion of our work and future work are described in the Section 6.

2. BACKGROUND

2.1 OpenStack Cloud OS

OpenStack is an open source project for the cloud computing platform that began in 2010. The project was started by Rackspace Hosting and NASA. Currently, OpenStack Foundation manages the project and more than 500 companies are participated in the project, including AMD, Cisco, Dell, Google, IBM, Intel, Oracle, VMware, etc. Updated version of OpenStack platform is released around every six-months under the terms of the Apache License.

When classifying Cloud computing according to service and deployment models, OpenStack can be classified as an infrastructure as a service (IaaS) for public, private, and hybrid cloud. OpenStack is composed of several lower branches of projects including Nova, Glance, Keystone, Neutron, etc. Fig. 2 shows the conceptive architecture of OpenStack projects. Our interest in OpenStack for this paper is focused on compute and network services

2.2 Data Distribution Service

DDS is a publish/subscribe based machine-to-machine (M2M) standard, approved by the Object Management Group (OMG) , that enables real-time, high-speed, and scalable data communication between publishers and subscribers. Many of todayâĂŹs systems are heterogeneous, mixture of different hardware, operating systems, programming languages, and development environments. In case of our TOC system, the weapon systems are heterogeneous. For example, the missile launch system uses Windows based OS with Java applications while the early warning system uses Linux based OS with C++ applications. In this scenario, integrating these two systems requires standard middleware APIs such as DDS, CORBA, and JMS.

DDS is widely used in many kinds of systems such as aerospace, military, traffic control, simulation, and medical

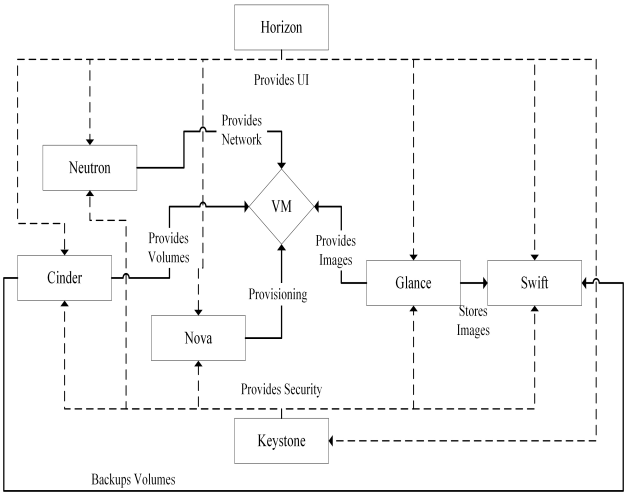


Figure 2: Conceptive architecture of OpenStack projects

system. Recently, there is a movement of adopting DDS to Internet of Things (IoT) system especially for Industrial Internet of Things (IIoT). RTI Corporation, which is DDS development Company, already participates in the Industrial Internet Consortium (IIC), and according to the market study conducted by Appinions in July 2014, RTI has been ranked the most influential industrial IoT Company [15]. This implies that DDS will take more important role in the IoT field.

2.3 GPU Virtualization

General Purpose computing on Graphic Processing Unit (GPGPU) is nowadays a common technique to boost up parallel applications that are traditionally processed by CPUs. GPGPU has been used in many scientific computing applications including molecular modeling, bioinformatics, financial computing, and digital signal processing. In particular, applications involving vector or matrix operations can gain large benefits from GPGPU programming. In our TOC system, algorithm for decision making, Montel Carlo simulation, and Kalman filter algorithm require GPGPU based system to support high-performance operational control in the combat situation. However, using GPUs in the virtual machine needs GPU virtualization techniques to support GPGPU programming on the guest OS. Currently, there are about three approaches for the GPU virtualization, PCI pass-through [10], remote-API, and hardware supported virtualization [4].

3. PROPOSED SYSTEM

Our TOC system has two major components, tactical and operational control system and weapon system. Tactical and operational control system includes combat control servers, tactical console servers, controller server, and data link servers. The combat control servers manage tactical and operational information and perform critical decision making during the combat situation. Tactical console servers provide visual information to the managers who command operational actions. Controller server is OpenStack controller server which manages the entire TOC system related to OpenStack components. Data link servers handle connection between internal/external weapon devices. Weapon system is a collage of multiple weapon device software such as missile launch and radar tracking software. Fig. 3 shows an overall architecture of the TOC system.

Every server in tactical and operational control system, except the controller server, is also a compute node of OpenStack Cloud platform. Each server has nova-compute, KVM hypervisor, and nova-network packages. With nova-compute and KVM hypervisor, the virtual machine is created in the target server. For example, weapon handling software which controls multiple weapons in combat situation is installed in virtual machine that placed at the combat control server. Every virtual machine in the TOC system communicates through DDS middleware. To support high speed data exchange between virtual machines, local network of the system is composed of 10Gbps Ethernet and QDR Infiniband network

3.1 Weapon System

Weapon system composed of multiple servers that are compute nodes of OpenStack. Weapon software related to SAM in the combat situation such as radar software, missile launch software, satellite software, and early warning software are installed in the virtual machines that are placed at servers. As same as the tactical and operational control system, DDS middleware is installed in the virtual machines.

3.2 Tactical and Operational Control System

Tactical and operational control system is the main component of the TOC for SAM system. The system manages the weapon systems and make control decision in the combat situation. Combat control servers are central part of the tactical and operational control system. Combat control software including sensor handling, air picturing, Threat Evaluation and Weapon Assignment (TEWA), weapon handling, weapon tactical, and basic utility software are installed in the virtual machines that placed at combat control servers. In addition, every server is also the compute node of OpenStack Cloud that nova-compute and nova-network are installed as OpenStack packages.

Some parts of combat control servers have special purpose virtual machines such as decision making in the combat situation, Kalman filtering for radar tracking, and Monte Carlo simulation to support probabilistic simulation. These servers have high-end GPUs to provide GPGPU features to the virtual machines and PCI pass-through and remote-API techniques are both implemented for the GPU virtualization. Tactical console servers include tactical multi-media interfaces (MMI) to support visualized monitoring over the TOC system and data link servers act as link handler between servers.

Controller server is a single server which manages the entire TOC system related to OpenStack features. Controller server has many OpenStack components such as keystone for authentication, glance for the guest OS image management, horizon for OpenStack dashboard support, heat for orchestration over the virtual machines, ceilometer for the server resource monitoring, and nova related packages for the virtual machine management. Because virtual machines are not created in this server, there is no KVM hypervisor. In addition, controller has only one network interface for an OpenStack management purpose while other servers have two network interfaces, one for OpenStack management and the other for data exchange between virtual machines. Every virtual machine in the tactical and operational system has DDS middleware to support high-speed and reliable communication between the virtual machines.

3.2.1 System Network

Neutron and nova-network are network services for OpenStack. In an earlier version of OpenStack, nova-network was the only network service for the virtual machine networking. From the OpenStack version Folsom, new network service project called Quantum has been released and from the version Havana, project name has changed to Neutron, which provides Software Defined Network (SDN) as a network service [1]. Despite of many difference between SDN and legacy network services, our interest for OpenStack network services focused on the floating IP range for the virtual machines. Legacy network, which is nova-network, assigns a fixed IP to the virtual machine. Main role of nova-network is to support Linux bridge network between the virtual network interface in the virtual machine and the physical network interface in the compute node. However, this fixed IP can only be used inside the Cloud system. To access

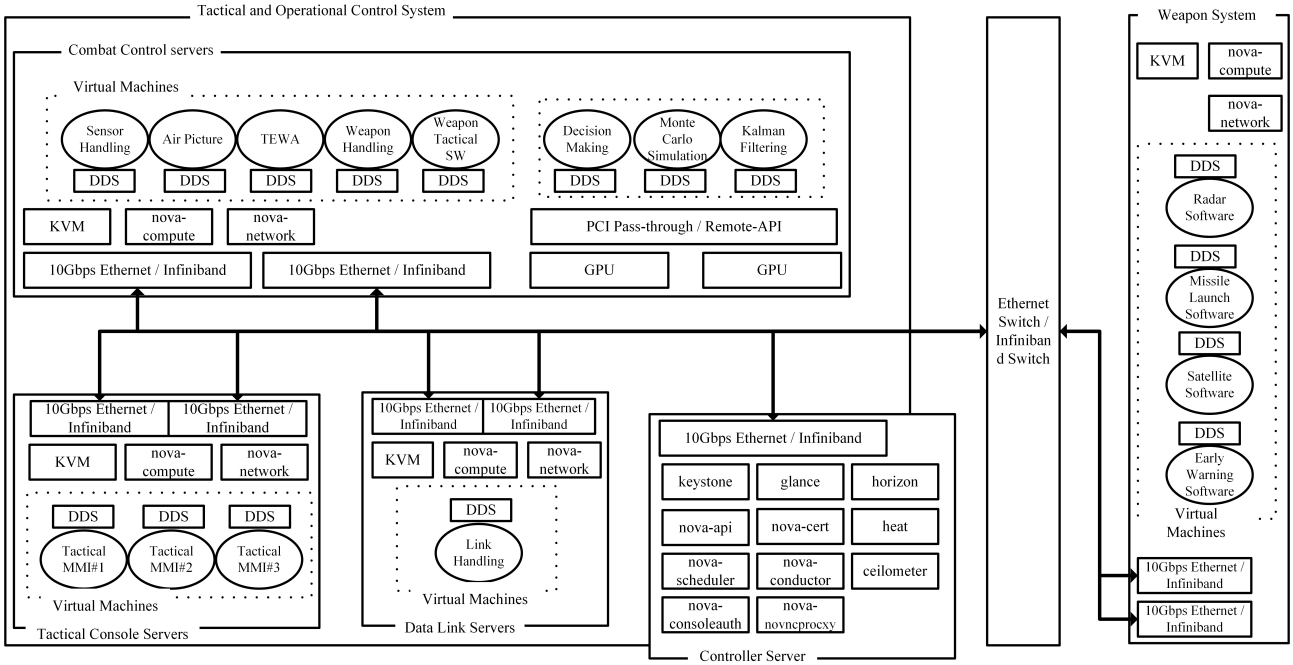


Figure 3: TOC System Architecture

the guest OS from the outside system, a floating IP should be assigned to the virtual machine. When nova-network is used, the range of floating IPs is same as that of the servers where OpenStack is installed. This limitation occurs a huge problem when the cloud system is composed of thousands of servers.

Neutron eliminates this limitation with SDN concept. When Neutron is used, network control software that was installed in the network devices are all programmed into the independent software. With Neutron, we can observe several new things such as ML2 plugin, L2 Agent, and OpenvSwitch that were not visible when nova-network is used. Because every network control software is configurable and programmable, the range of floating IPs is no longer limited by the range of the host servers. Although many of todayâĂŹs public cloud service deployed SDN for Cloud networking in the data center, using SDN for networking in private cloud service, like our TOC system, can be inefficient especially when the cloud system only requires small number of virtual machines, particularly less than 1,000.

3.2.2 Virtual Machine Communication

DDS middleware provides communication between the virtual machines. DDS is composed of Data-Centric Publish-Subscribe (DCPS) and Data Local Reconstruction Layer (DLRL) [17]. DCPS is the lower part of APIs layer that enables applications to communicate topic data with other applications. Meanwhile, DLRL is the upper part of the layer that allows integration between applications and DDS interfaces. Communication between applications are separated by the domain that applications with same domain only can exchange their data through the DDS. Fig. 4 shows overall architecture of DDS, especially includes DCPS entities. Topics are basic unit of data to be published and subscribed. Domain participants are entry points for data access. Publishers contain Data-writers to perform the write operation and subscribers have Data-reader to access data. There are two ways of accessing data through the subscriber, wait-based data access and listener-based data access. The wait-based approach blocks participant until data change occurs, which is a synchronous approach. However, the listener-based approach notifies participant whenever data changes, which is an asynchronous approach. These DCPS entities, topics, data-writers, data-readers, are configurable with quality of service (QoS) policies that consider deadline, resource limitation, reliability, durability, etc.

3.2.3 GPGPU Support

We used two different GPU virtualization techniques to support GPGPU for acceleration of the decision making. Firstly, PCI pass-through technique binds physical GPUs to the virtual machine through the hypervisor. This technique requires IOMMU support from CPU. Fig. 5 shows an architecture of the PCI pass-through for GPUs. Attaching GPUs to the guest OS requires following steps, removing GPUs from the hostâĂŹs PCI interface, binding GPUsâĂŹ vendor and device ID to the PCI stub driver, and assigning GPUs on the virtual machine creation time. In case of OpenStack, vendor and device ID of GPUs are defined in Nova configuration file with pci-passthrough-whitelist and pci-alias options. OpenStack also supports scheduling of GPUs with PCIPassthroughFilter option in nova-scheduler of the controller node [2]. When a virtual machine creation is requested, GPUs are pre-allocated to the virtual machine flavors and user can access GPUs from the guest OS as same as accessing from the host OS.

Advantage of PCI pass-through technique comes from the direct access of GPUs from the guest OS PCI interface. User can use every feature of GPUs from the guest OS as it was in the host OS, and the performance of GPU applications

shows almost near performance as that of native GPU. However, there is a critical limitation which is a lack of scalability. GPUs that are bound to the virtual machine cannot be shared with other virtual machines which compromises inefficient use of GPU resources. Unlike CPUs, GPUs are used in small fraction of time compared to the idle time of CPUs. Increasing idle time of GPUs in an entire system is crucial, especially for the Cloud service provider due to the high cost of GPU resources. In case of Amazon EC2 public Cloud service, which is the most popular IaaS Cloud service today, virtual machine with GPUs are about one dollar per hour more expensive than the same flavor of virtual machine without GPUs [12]. In conclusion, sharing GPUs resource among virtual machines are an important issue in GPU the virtualization technique and this limitation brought remote-API virtualization technique.

Secondly, Remote-API technique allows using GPUs in remote host through the network. Main idea of remote-API techniques is capturing API calls from GPU applications, then send these API calls to BackEnd module in the host server side which has GPUs. After GPU driver and GPU execute API calls, the result returns back to GPU applications through network or loopback interface. Fig. 6 shows the general architecture of remote-API technique. When GPU application starts, FrontEnd module captures API calls with the wrapper library. Then FrontEnd module sends captured API calls to BackEnd module through the network interface. Because FrontEnd and BackEnd modules are separated, GPU application can be executed even there is no GPU in the host where the application is running. After BackEnd module receives API calls, it delivers API calls to GPU driver. When GPU finishes execution for API calls, the result returns back to FrontEnd module and to GPU application. Main advantage of remote-API technique is a separation of FrontEnd and BackEnd modules that enables GPU applications to use GPUs in the remote hosts. This helps an efficient use of GPU resources when the system is distributed and not all hosts have GPUs for computing resources. Primary target system of remote-API was a large distributed system with small number of GPUs so that the applications in every hosts can use GPUs logically through the network interface. Nowadays the virtualized system such as Cloud computing has huge interests in the remote-API technique. With remote-API, the virtual machine can access GPUs in the server host through the virtualized network interface.

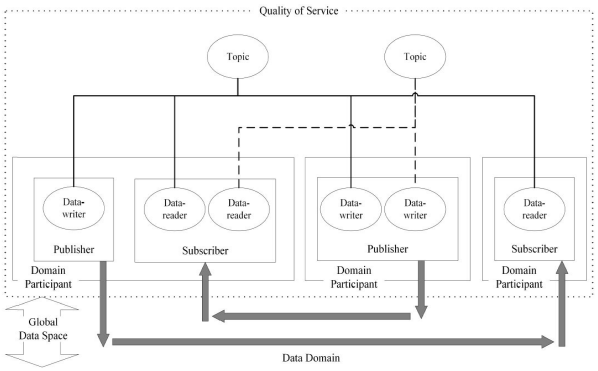


Figure 4: Overall Architecture of DDS

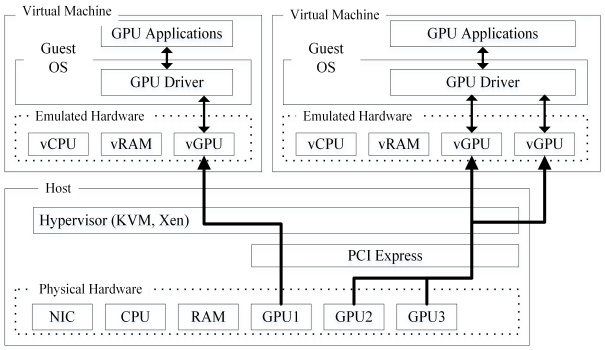


Figure 5: PCI Pass-through Architecture

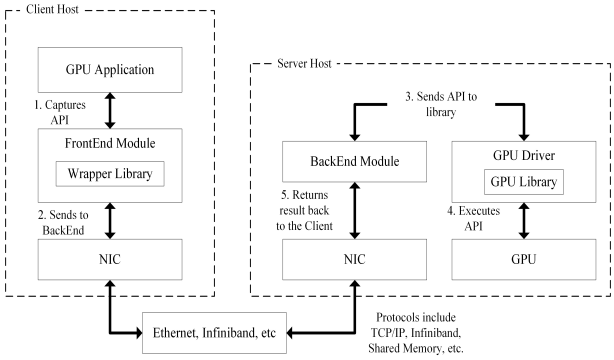


Figure 6: Remote-API Architecture

Our interest in GPU virtualization also focuses on the remote use of GPUs among the TOC system. However, because data communication between GPU application and GPU memory is processed through the network interface, performance of GPU applications highly depends on the network status. In addition, GPU driver is installed in the host server side, not in the client side, which limits GPU features from the client side OS due to the invisibility of GPU through the PCI-e interface. Despite the limitation of GPU features and performance issue, remote-API is still attractive GPU virtualization technique that allows remote use of GPUs in the distributed system. There are several remote-API methods such as gVim [11], gVirtus [24], vCUDA [19], and rCUDA [20] [13]. As far as we know, rCUDA is the only updated remote-API technique which supports CUDA version 7.0. Table 1 describes difference between these remote-API techniques.

Table 1: Comparison between Remote-API methods

	gVim	**gVirtus**	**vCUDA**	**rCUDA**
Version	None (2009)	Beta3 (2011.11)	None (2012.6)	v15.07 (2015.7)
S/W Available	No	Yes	No	Yes
GPGPU Model	CUDA1.1	CUDA3.2 OpenCL1.1	CUDA3.2	CUDA7.0
Network Channel	XenStore	TCP/IP (vm-Socket)	VMRPC (Shared memory)	TCP/IP Infini-band

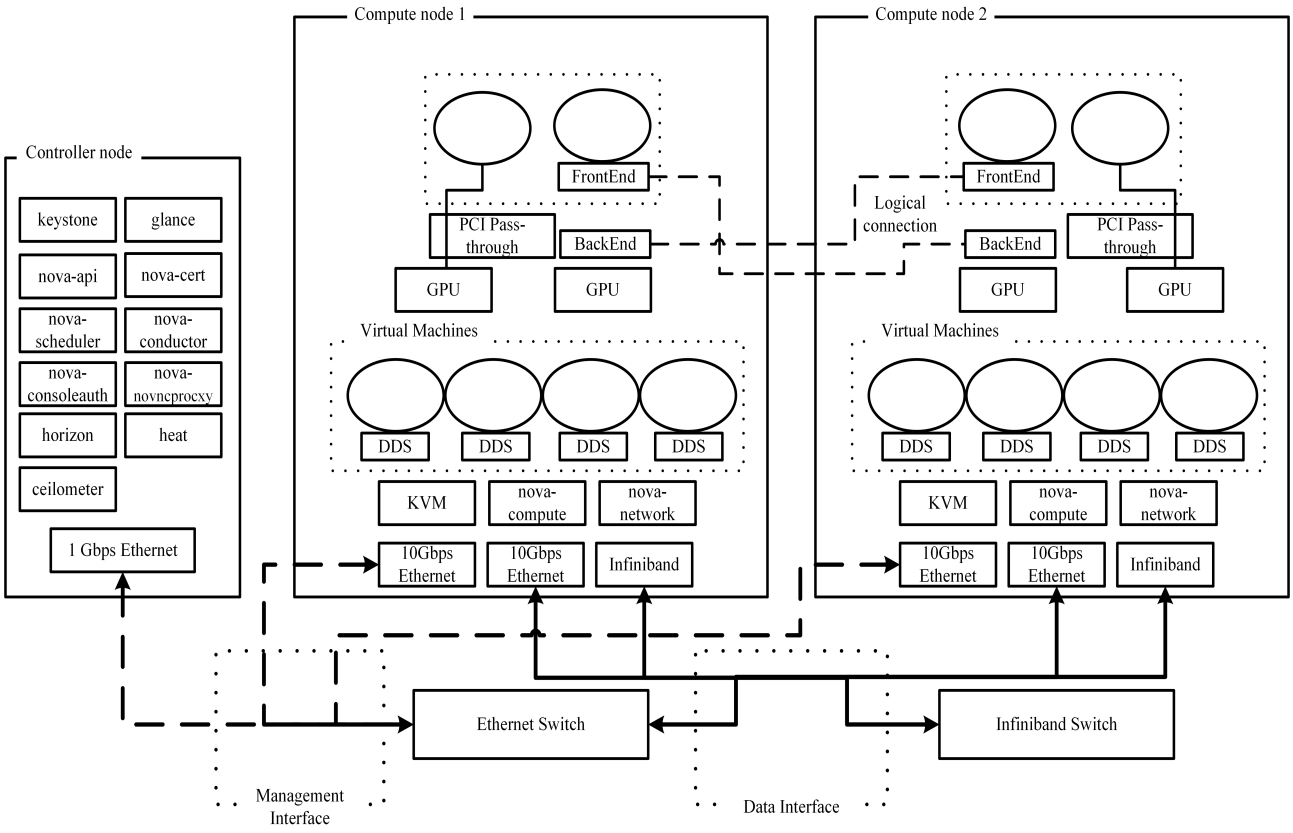


Figure 7: Experimentation System Architecture

4. EXPERIMENTAL EVALUATION

Our experiments are categorized into two different evaluation. First, we evaluate the performance of DDS middleware in virtualized and non-virtualized environments with 10Gpbs Ethernet and QDR Infiniband network. In addition, we evaluate the scalability of DDS middleware with eight virtual machines across two distributed servers. Second, we analyzed the performance of general GPU applications with two different GPU virtualization techniques, PCI pass-through and remote-API, on our experimentation system.

4.1 Experimental Setup

To evaluate the performance of the TOC system, we implemented another system for the experimentation. Due to the military security reason, it is forbidden to expose the actual TOC system specification to the public. So we implemented a reduced size TOC system for the research and experimental purpose. The system composed of one PC for the controller node and two servers for the compute nodes. The PC has Intel Core i7 CPU and 8GB RAM while each server has two Intel Xeon E5 CPUs, 64GB RAM, and two high-end NVIDIA Tesla K20 GPUs. Infiniband that we used for the experimentation is QDR Infiniband that theoretically shows 40Gbps bandwidth on Infiniband protocol while shows 10Gbps on IP over Infiniband (IPoIB) configuration. Ubuntu 14.04 is used in every host OS with Linux kernel version 3.19.0-26. Fig. 7 shows an overall architecture of the experimentation system. As same as the actual TOC system, we used OpenStack Cloud OS to support virtualized environment. We have one controller node with keystone, glance, nova, horizon, heat, and ceilometer installed and two compute nodes with KVM hypervisor, nova-compute, and nova-network.

For the management interface, used to deliver commands between OpenStack components, is composed of 1 Gbps Ethernet interface from the controller node and 10 Gbps Ethernet interfaces from compute nodes. Data interface provides data exchange between virtual machines. This is the most important interface to DDS middleware because the communication through DDS will use data interface. As result, data interface is composed of two high-speed networks, 10Gbps Ethernet and QDR Infiniband. However, nova-network does not support Infiniband protocols for the internal network interface, so we deployed Infiniband interface into a virtual Ethernet interface by using Ethernet on IP over Infiniband (E-IPoIB) [6]. Because of additional interface for the IPoIB, E-IPoIB shows lower bandwidth than that of IPoIB which is a theoretically up to 10Gbps.

In the system, each compute node has two NVIDIA Tesla K20 GPUs for the high performance GPGPU computing. One of GPUs is bound to the virtual machine through PCI pass-through and the other is bound through remote-API, especially rCUDA. rCUDA is the only remote-API method which supports recent CUDA version with other important GPGPU libraries such as CUBLAS, CUDNN, CUFFT, CURAND, and CUSPARSE. With rCUDA, the virtual machine can use GPUs in both compute nodes that are not

bound to the specific virtual machines through PCI pass-through. When the virtual machine executes GPU applications, rCUDA uses management interface for API and data exchange between FrontEnd and BackEnd modules. For this reason, management interface of compute nodes can be either 10Gbps Ethernet or virtual Ethernet interface with E-IPoIB.

To measure the performance and scalability of DDS middleware, we used RTI Connext DDS Professional 5.1.0 [22] and RTI PerfTest 5.1.0.9 [21]. For the performance measurement, we set the following four different environments, non-virtualized 10Gbps Ethernet, non-virtualized QDR Infiniband with E-IPoIB, virtualized 10Gbps Ethernet, and virtualized QDR Infiniband with E-IPoIB. Virtualized environments are composed of m1.large type virtual machine in OpenStack Cloud which uses two virtual CPUs, 8GB RAM, and 80GB disk with an Ubuntu 14.04 for the guest OS. From DDS, each publisher sends 100,000 messages with message queue size 50, and message length from 1 byte to 10bytes with multicast option enabled. DDS performance is measured with average latency of single message, throughput of the subscriber, and standard error of latency in the single test.

To evaluate DDS more precisely, we set the interval time between messages sent from publisher to subscriber. For the latency driven evaluation, we set the interval time to 1ms to check the latency difference between 10Gbps and E-IPoIB QDR Infiniband. For the throughput driven evaluation, the interval time is set to 0.1ms to check the throughput difference between the two networks.

Scalability of DDS in virtual machines is measured through eight virtual machines that are separated into four each. Each compute node has up to four virtual machines and we measured the latency and throughput of publishers and subscribers with one-to-one to four-to-four matching.

For the performance measurement of GPU applications, we used Rodinia 3.0 benchmark suite [5] which is generally used to measure the performance of real-world applications written in GPGPU programming languages including CUDA and OpenCL. Within the Rodinia suite, we chose 8 GPU applications written in CUDA, Backprop, BFS, CFD, Hotspot, LUD, NN, NW, and Pathfinder. Table 2 shows description of GPU programs used in Rodinia suite.

Table 2: GPU Benchmark Program Description

Name	Description
Bacprop	Back Propagation which is a machine learning algorithm that used in neural network
BFS	Breadth-First-Search (BFS) is well known graph traversal algorithm
CFD	CFD is a volume solve program for 3D Euler equations for compressible flow
Hotspot	Hotspot is used to estimate the processor temperature
LUD	LU Decomposition is well known algorithms to solve linear equations
NN	K-nearest neighbors is popular algorithm in pattern recognition for classification and regression
NW	Needleman-Wunch is optimization method that used for DNA sequence alignments
Pathfinder	Pathfinder uses dynamic programming to find a path in 2D grid with the smallest accumulated weights

4.2 Evaluation Result

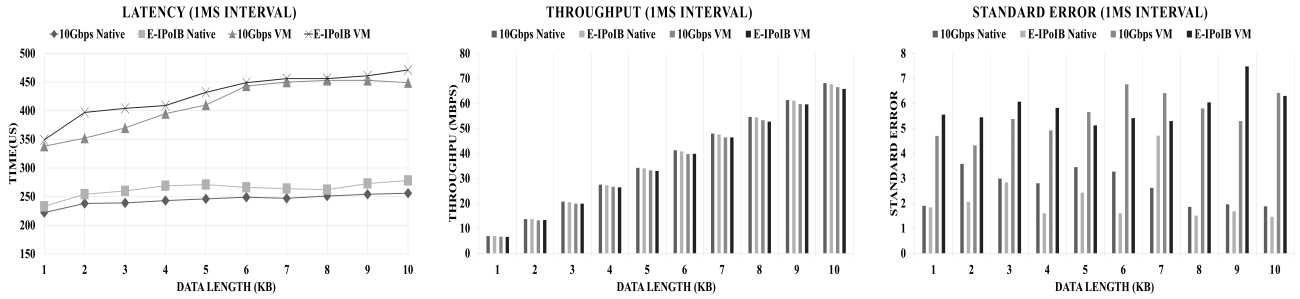
4.2.1 DDS performance evaluation results

Fig. 8 shows the performance evaluation results of DDS with 1ms message interval. In case of our TOC system, actual message interval period required in the system is between 100ms and 1000ms which is much slower than 1ms. As result, we concluded that testing with 1ms message interval is sufficient enough to show stable latency over the TOC system. From the Fig. 8a, average latency of DDS on 10Gbps Ethernet is 5% lower than that of E-IPoIB Infiniband. Between non-virtualized and virtualized, average latency in non-virtualized environments show 65% lower than that of virtualized one due to the increased number of layers from the virtualized network interface. From the Fig. 8b, throughput of subscriber linearly increases when data length of message gets higher which is reasonable result considering that messages are published with multicast option enabled. From the Fig. 8c, standard error of every test is below 10 which is stable enough in case of our TOC system requirement.

Fig. 9 shows the performance evaluation results of DDS with 0.1ms message interval. Compared to the 1ms interval message, 0.1ms interval drives the TOC system network throughput into the limitation. With the 0.1ms interval test, we purposed to check the negative effect of blocked messages over the entire TOC system. From the Fig. 9a, average latency of DDS in non-virtualized environment is 41% lower than that of virtualized one until the message length reaches 5 bytes. However, after the 6 bytes message length, average latency non-virtualized environments is 260% lower than that of virtualized one. The reason of this abnormal increase of latency results in virtualized environment is from the increased number of blocked messages. When the message queue of publisher is full, messages are blocked until the queue gets empty. The latency of this blocked message is over 30,000us which is about 100 times slower than the stable latency. Table 3 shows the maximum latency of 0.1ms interval time results. From the Table 4 and Fig. 9c, we can observe that standard error also dramatically increases when the messages are blocked. From the Fig. 9b, we can observe that throughput of virtualized environments reaches 1Gbps than decreases after message length of 5 bytes. The reason of this throughput limitation comes from virtio, which is I/O virtualization layer used in KVM hypervisor [23]. When virtio is used, there is only one queue for each network receive (Rx) and transmit (Tx) in the virtual machine. This default option limits network throughput of virtual machine around 1Gbps. However, we believe that it is not a reliable approach to increase the number of queues for Rx and Tx for the single virtual machine due to the load balance issue of traffics between virtual machines. We will re-discuss this problem in further research.

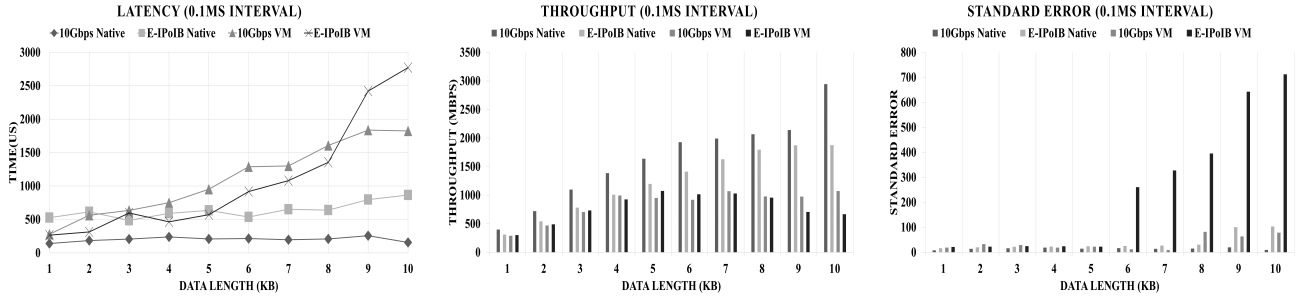
4.2.2 DDS scalability evaluation results

Fig. 10 shows the evaluation results of DDS scalability. To get a stable results, we configured interval time between messages to 1ms. From the Fig. 10a, latency of DDS gets lower when data length of the message increases. Between 10Gbps Ethernet network and E-IPoIB Infiniband network, E-IPoIB Infiniband shows lower latency until data length of the message reaches 5 bytes. However, from the 6 bytes, 10Gbps shows better latency than that of E-IPoIB Infiniband net-



(a) Latency (b) Throughput (c) Standard Error

Figure 8: DDS Performance Evaluation Results with 1ms Message Interval



(a) Latency (b) Throughput (c) Standard Error

Figure 9: DDS Performance Evaluation Results with 0.1ms Message Interval

work. This result implies that DDS should be carefully configured considering the number of publisher and subscribers, and data length of the messages. From the Fig. 10b, we can also observe that the throughput of each subscriber increases when data length of messages gets higher. These results can be different when the multicast option for DDS is turned off. In this experimentation, we set the multicast option to get the best results from DDS and also it is a general approach to set multicast option when using DDS with multiple publishers and subscribers.

Table 3: Maximum Latency Result

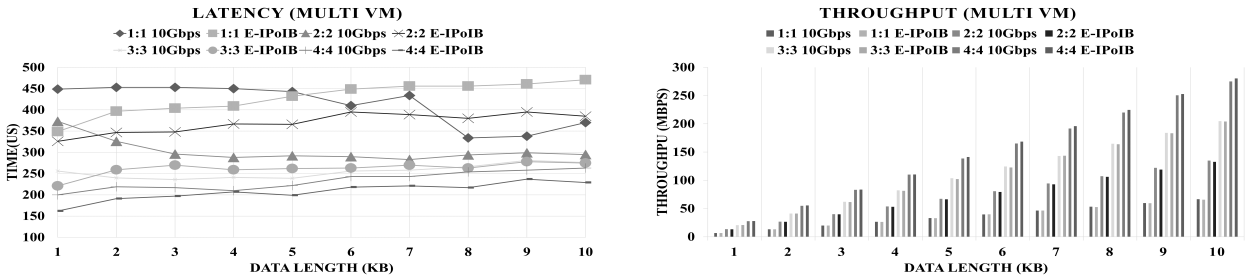
Size(KB)	10Gbps Native	E-IPoIB Native	10Gbps VM	E-IPoIB VM
1	821	1024	2094	893
2	800	1597	3146	939
3	864	1169	2881	1320
4	889	1391	1984	1011
5	1025	1203	2477	1008
6	1217	1311	3943	32217
7	1155	1220	2397	32900
8	1213	1837	33461	33221
9	1238	31442	35500	33598
10	1128	31249	34901	33964

Table 4: Standard Error Result

Size(KB)	10Gbps Native	E-IPoIB Native	10Gbps VM	E-IPoIB VM
1	8.08	16.97	19	21.85
2	13.93	19.84	33.04	22.43
3	15.94	22.09	28.86	24.69
4	19.09	23.21	18.84	24.34
5	14.61	24.18	22.64	22.4
6	16.83	25.52	12.24	260.59
7	13.84	27.14	9.36	327.4
8	15.24	30.76	81.96	395.55
9	19.95	101.04	63.55	642.78
10	9.77	103.61	78.76	712.27

4.2.3 GPU benchmark programs evaluation results

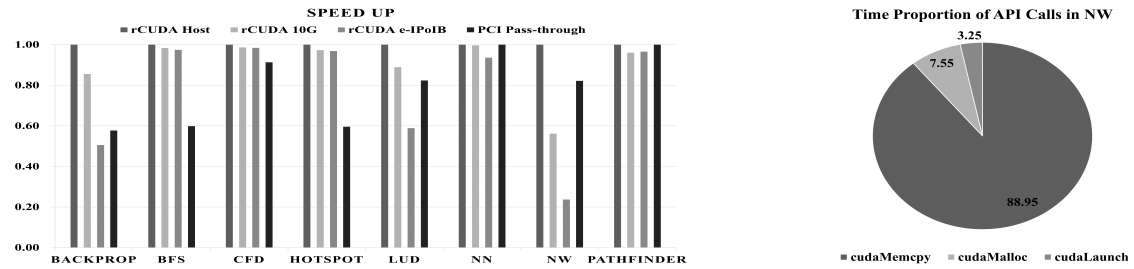
Fig. 11 shows the performance evaluation results of GPU benchmark programs in Rodinia suite. Fig. 11a, shows normalized execution time based on the results of rCUDA using localhost GPUs through loopback interface. From the result, we can observe that PCI pass-through does not always show better performance than that of rCUDA. The reason of performance gap between PCI pass-through and rCUDA comes from pre-load of runtime libraries into the main memory. When rCUDA is used to execute GPU applications, the runtime library is a chunk of wrapper libraries which is pre-loaded on the main memory. Because whole library is already loaded, processing time of API calls is faster than that of original approach. However, this advantage is removed when there is a lot of memory copy between GPU application and GPU memory. From the Fig. 11b, proportion of execution time for the memory copy API calls in NW benchmark is higher than 80%. In result of NW benchmark, the average execution time of the NW benchmark using remote GPUs through the networks is 128% higher than result of using the local GPUs. In contrast, the average execution time of other 7 benchmarks using remote GPUs is only 7% higher than that of using local GPUs.



(a) Latency

(b) Throughput

Figure 10: DDS Scalability Evaluation Results



(a) Speed Up

(b) API Calls

Figure 11: GPU Benchmark programs Performance Evaluation Results

5. RELATED WORK

5.1 DDS over the Cloud environment

There are several researches try to adopt the DDS middleware in the Cloud environment. Corradi, A., et al. [7] introduced Cloud monitoring and management architecture with the DDS standard. They also used an open source tool called Distributed Architecture for Resource manaGement and mOnitoring in cloudS (Dargos) [18] to implement their own management framework related to OpenStack Cloud OS. Similar to [7], An, K., et al. [3] also introduced a resource monitoring middleware in the Cloud environment with DDS. However, main purpose of [7], [3] is to introduce monitoring/management of Cloud infrastructure with DDS while our work targets to build an entire Cloud system with DDS as a data communication middleware between virtual machines.

5.2 GPU Virtualization techniques on Cloud

Performance evaluation of GPU virtualization techniques such as PCI pass-through and remote-API is done by many previous researches. Crago, S., et al. [8] introduced OpenStack Cloud platform with remote-API method called gVirtus. Younge, A., et al. [26] evaluated PCI pass-through on Xen hypervisor with SHOC [9] benchmark suite. Walters, J. P., et al. [25] evaluated performance of PCI pass-through in several hypervisors such as KVM, Xen, VMware ESXi, and LXC with SHOC benchmark suite. Jun, T., et al. [14] evaluated the performance of PCI pass-through and rCUDA remote-API on the OpenStack Cloud platform with SHOC and Rodinia benchmark suites. However, evaluation of GPU virtualization techniques on our work focuses on the performance of PCI pass-through and rCUDA on the OpenStack Cloud OS with two different high-speed networks, 10Gbps Ethernet and QDR Infiniband.

6. CONCLUSION AND FUTURE WORK

TOC for SAM system is an important platform that controls weapon systems related to the SAM in the combat situation. Previous approaches of TOC system had several limitations such as unable to support heterogeneous environments between weapon systems, inefficient computing resource management, and lack of GPGPU based system to provide high-performance decision making. In this work, we proposed a novel HPC supported mission-critical Cloud architecture as TOC for SAM system with OpenStack Cloud OS, DDS middleware, and GPU virtualization techniques. We also implemented an experimental TOC system to evaluate performance DDS, scalability of DDS, and performance of GPU virtualization techniques over two different networks, 10Gbps Ethernet and QDR Infiniband. From the evaluation result, we conclude that efficiency of DDS for communication between virtual machines highly depends on characteristics of published messages such as data length, message queue size and interval time between messages. We also observed that the number of blocked messages critically affects the increase of average latency by checking standard errors of latency results. In conclusion, we deployed an actual TOC for SAM system for Agency of Defense Development with considering evaluation results of experimental system.

In future, we plan to expand our TOC system into general purpose Cloud platform such as IoT platform. Recently, DDS middleware is used to support communication between sensors in IoT platform. Similar to the TOC system, DDS middleware handles data exchange of sensors while GPGPU system provides analysis of big data gathered from IoT sensors and actuators with machine learning algorithms.

7. ACKNOWLEDGMENTS

The authors would like to thank the Agency of Defense Development and LIG Nex1 Corporation for their support that has contributed to the research results reported within this paper.

8. REFERENCES

[1] D. Abramson. Intel virtualization technology for directed I/O. In *Intel technology journal*, 2006.

[2] Amazon. Amazon ec2 retrieved from http://aws.amazon.com/ko/ec2. 2015.

[3] K. An, S. Pradhan, F. Caglar, and A. Gokhale. A publish/subscribe middleware for dependable and real-time resource monitoring in the cloud. In *Proceedings of the Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management*, 2012.

[4] Appinions. Internet of Things : An Industry Influence Study Retrieved from http://dj.appinions.com/iot-july-2014, year = 2014.

[5] S. Che, M. Boyer, J. Meng, D. Tarjan, J. Sheaffer, S.-H. Lee, and K. Skadron. Rodinia: A benchmark suite for heterogeneous computing. In *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, 2009.

[6] J. Chu, , and V. Kashyap. Transmission of ip over infiniband (ipoib) no. rfc 4391. 2006.

[7] A. Corradi, L. Foschini, J. Povedano-Molina, and J. Lopez-Soler. DDS-enabled Cloud management support for fast task offloading. In *Computers and Communications (ISCC), 2012 IEEE Symposium on*, 2012.

[8] S. Crago, K. Dunn, P. Eads, L. Hochstein, D.-I. Kang, M. Kang, D. Modium, K. Singh, J. Suh, and J. Walters. Heterogeneous Cloud Computing. In *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*, 2011.

[9] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter. The Scalable Heterogeneous Computing (SHOC) benchmark suite. In *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, 2010.

[10] P.-C. G. Omg data-distribution service: Architectural overview. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, 2003.

[11] G. Giunta, R. Montella, G. Agrillo, and G. Coviello. A GPGPU Transparent Virtualization Component for High Performance Computing Clouds. In *Euro-Par 2010-Parallel Processing*, 2010.

[12] V. Gupta, A. Gavrilovska, K. Schwan, H. Kharche, N. Tolia, V. Talwar, and P. Ranganathan. GViM: GPU-accelerated virtual machines. In *Proceedings of the 3rd ACM Workshop on System-level Virtualization for High Performance Computing*, 2009.

[13] A. Herrera. NVIDIA GRID: Graphics Accelerated VDI with the Visual Performance of a Workstation. In *Nvidia Corp*, 2014.

[14] T. J. Jun, M. H. Y. Van Quoc Dung, D. Kim, H. Cho, and J. Hahm. GPGPU enabled HPC Cloud Platform based on OpenStack Retrieved From http://sc14.supercomputing.org/sites/all/themes/sc14/files/archive/tech_poster/poster_files/post269s2-file3.pdf. 2014.

[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. In *ACM SIGCOMM Computer Communication Review*, 2008.

[16] OpenStack. OpenStack Cloud Retrieved from http://www.openstack.org. 2015.

[17] OpenStack. Pci passthrough retrieved from https://wiki.openstack.org/wiki/Pci_passthrough. 2015.

[18] J. Povedano-Molina, J. M. Lopez-Vega, J. M. Lopez-Soler, A. Corradi, and L. Foschini. DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds. In *Future Generation Computer Systems*, 2013.

[19] C. Reano, R. Mayo, E. S. Quintana-Orti, F. Silla, J. Duato, and A. J. Pena. Influence of InfiniBand FDR on the performance of remote GPU virtualization. In *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*, 2013.

[20] C. Reano, F. Silla, A. Pena, G. Shainer, A. Schultz, S.and Castello, E. Quintana-Orti, and J. Duato. Boosting the performance of remote GPU virtualization using InfiniBand connect-IB and PCIe 3.0. In *Cluster Computing (CLUSTER), 2014 IEEE International Conference on*, 2014.

[21] RTI. RTI Connext DDS Combined Latency and Throughput Performance Test Getting Started Guide Retrieved from http://community.rti.com/rti-doc/510/RTI_Performance_Test_5.1.0/doc/RTI_ConnextDDS_PerformanceTest_GettingStarted_5.1.0.pdf, year = 2014.

[22] RTI. RTI Connext DDS Professional Getting Stated Guide Retrieved from http://www.rti.com/eval/rtidds510/RTI_Connext_DDS_Professional_GettingStarted.pdf. 2013.

[23] R. Russell. virtio: towards a de-facto standard for virtual I/O devices. In *ACM SIGOPS Operating Systems Review*, 2008.

[24] L. Shi, H. Chen, J. Sun, and K. Li. vCUDA: GPU-Accelerated High-Performance Computing in Virtual Machiness. In *Computers, IEEE Transactions on*, 2012.

[25] J. P. Walters, A. J. Younge, D. I. Kang, K. T. Yao, M. Kang, S. P. Crago, and G. C. Fox. GPU Passthrough Performance: A Comparison of KVM, Xen, VMWare ESXi, and LXC for CUDA and OpenCL Applications. In *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, 2014.

[26] A. Younge, J. Walters, S. Crago, and G. Fox. Evaluating GPU Passthrough in Xen for High Performance Cloud Computing. In *Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*, 2014.