Improving Energy Efficiency and Impairing Environmental Impacts on Cloud Centers by Transforming Virtual Machine into Self-Adaptive Resource Container

Siva Shanmugam[1] and Sriman Iyengar[2]
[1]Assistant Professor, School of Computer Science, India
[2]Information Technology, Sreenidhi Institute of Science and Technology, India

Abstract: *Enterprises are seeking on-demand computing models that can be employed with better utilization and reduced operational cost by remitting up to the users' needs; this brings up zero charges as zero demand exhibits because user demands are vary drastically over time. To reinforce dynamic resource provision, service providers have to maintain more computational resources than needed. Meanwhile, the IT sector has more apprehensions about the impact on the environment due to an increase in carbon dioxide emissions, higher electricity consumption and a growth in the electronic wastes from electronic components. Most of the research works focus primarily on the expertise required for providing the needed resources and not care on resource utilized which brings unsustainability. To achieve sustainable computing, unwanted installation of contemporary computational resources should be rolled up and better sharing options should be made available. This paper proposes new virtualization techniques which engage cloud services exclusively between host and guest operating environments. By doing so, this mechanism stands as the best crossover with other working engines and provide open service to execute any type of applications on it. Finally, the combination of cloud service and virtualization enables container features with efficient utilization factor. Most probably, a proper combination of these resources solves any computational issues so these two resource mechanism's always standing on the top of the change. This experiment analysis aims to compare the performance of container with virtual machine based container in an adopted infrastructure via cloud simulator. And the result of better efficiency metrics attained by virtual based container were explored and plotted.*

Keywords: *Container as a service, green cloud computing, energy efficiency, resource utilization.*

Received December 2, 2016; accepted March 26, 2017

1. Introduction

In the recent past, Internet becomes a significant locus to humans because all their requirements were fitted in it. The services running on internet have customers from all the corners; starting from communication mail to computational yell. In order to provide flaw less services, a computational setup need to be established for all corners which is highly impossible due to cost effective [12]. So IT renders services by duplicating the existing resource to get access on user environment under the name of virtualization started in 1960 onwards. In a short span of time virtualization attained several revolutions like Para and Full hardware assisted virtualization, where entire hardware resource isolated from single real machine to multiple environments.

In the mid of 1960-1970, the use of bandwidth, network traffic, demand on computational resource, extended infrastructure management is very low due to less customer connectivity to internet. Report [9] says only 1% of users around the globe were connected with internet during 1995-2000 Figure 1. Later, usage started increasing due to more user connectivity to internet to 75% (2015-2020*) leaving many challenging problems like bandwidth, congestion and other demands etc., So improvisation is required in terms of what type of resources provided to the users and how we are providing those resources (i.e.,) handling the resources as well deployment models.

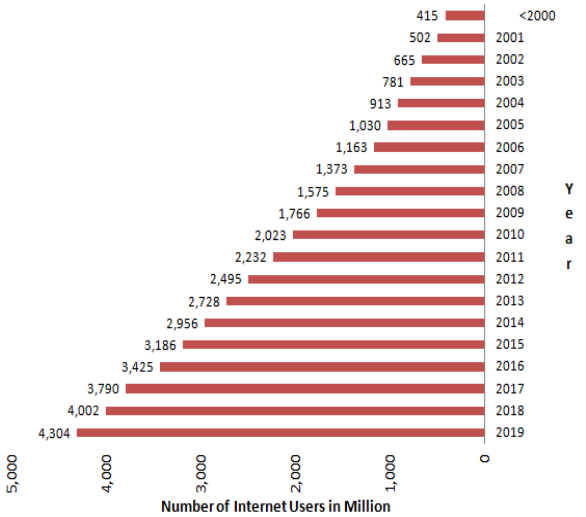


Figure 1. Internet users strategy.

2. Sense of Sustainable Computing

Earlier computing services are used to share the stored data between the systems over a private network (distributed system). Technology development in due course and need of demands changed this scenario as pay per use of computational resource services to the end users. This transformation is represented in Figure 2. After this number of hosting servers were crowded on provider side leading to more energy consumption and dissipation of CO_2. This stand as insecure to the IT sectors and to bring inevitability they introduce new methodologies on server consolidation named as container.

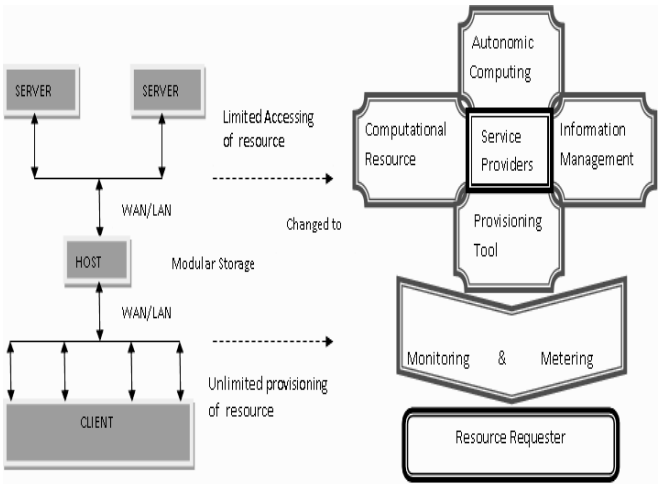


Figure 2. Traditional computing to modern computing.

3. Related Work

Several researchers investigated container and proposed algorithms to make container compatible with their application. Some of their works and the downside of containers are presented below.

Liu *et al*. [10] introduced container cluster system for solving scientific tool installation problems through Docker organization. He *et al*. [6] used to keep tools image in Docker host for later availability to Docker enabled machines to avoid bottleneck on installing tools from single machine. Zhang *et al*. [15] presented MPI library for containers which allow needed middleware tools to run on different containers. Whenever particular middleware tools requested by user on different container it automatically share the MPI files based on library content [11]. Celesti *et al*. [2] used container for IOT enabled devices to measure the performance and feasibility of applications inside the container. He used two types of container, for same sort applications runs on a container and another one is for user spaced request. Varghese performed several validation procedures on container and comes with bench mark [13] procedures to comply container for different applications.

Gerlach *et al*. [5] proposed a framework for container based libraries, which make applications to run parallel on Containers as well as automatically update the software patches available inside the containers. Abdelbaky *et al*. [1] develops a framework called C-port which bring transparency on deploying applications and also support ease migration between containers. Huang and Knottenbelt [7], once again developed one more library framework to support automation and parallelism in containers. Dhakate and Anand [3] introduced a new concept called Container As Service (CAAS), which apply container service inside the Virtual Machine (VM), so we can create virtual machines in an iterative manner. When comparing to other methodologies, this service based container may give better performance metrics.

4. Virtualization-Ancestry of Container

Virtual machines provide a sharing of hardware resources on guest OS through hypervisor irrespective of applications. We can create any number of guest OS on single host OS [9]. Instead of creating several guests OS, IT practice to have single host OS named as container, where all the applications are dumped on top. Even though it is a different layered working structure, applications run by virtualizing the resources. It requires additional binary libraries on host system to achieve isolations [14].

5. Problem Evaluation

Container is nothing but OS level virtualization initiating all the applications with single OS and along with additional middleware tools and MPI library files to validate. The problems identified by using containers on cloud are follows

- When a complexity trend started to execute on single host OS, then it's over burden to the system and it never support abundance of users. Sharing common middleware tools are not feasible and it requires varieties of intermediate tools to support but which is not feasible to run on common OS. Due to diverse of user's with diverse of requirements containers can't support rather than public cloud.

To overcome these drawbacks and improve VM functionalities on utilization factors, a hybrid model of light weight VM instance has been proposed and shown in Figure 3.

6. Proposed Model

A comparative of VM and container is shown in Figure 3, a difference of running on guest OS and host OS is clearly shown.

In Our proposed system, we included IAAS concept and PAAS concept in between host OS and Guest OS. By this, we are providing new behavioral functionality to virtualization life era.

- Isolation of application environments at any time with any VM is possible.
- Sharing common middleware requisites between instances and their users is possible.

7. Design Model

The overall working model of proposed system is shown in Figure 4. The proposed model is targeted to achieve adaptive container that support VM instance with splitting of processor based on application requirement. The detailed working structure was follows.

7.1. Host Status Monitor Module

Based on application size and SLA agreement, physical machines are split into consecutive virtual machines. In order to avoid system imbalance, application which depresses the performance of VM or VM which running with less executional power is to be identified and capacity should be extended. This process is done here and carried by maintaining threshold value on both host as well as VM instance while initiated.

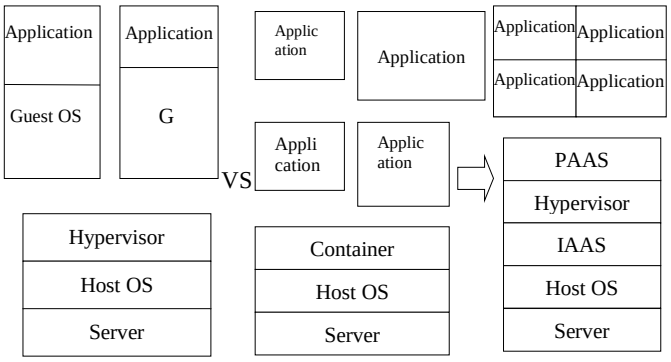


Figure 3. Proposed model of extended virtualization.

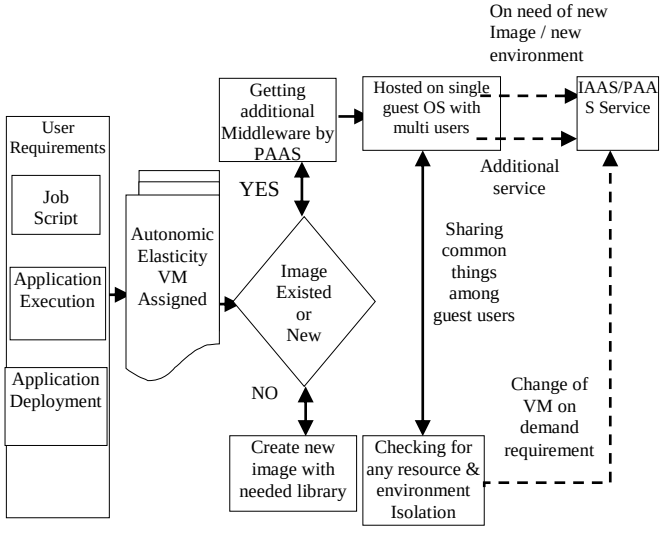


Figure 4. Architecture design of self adaptive virtualization.

Table 1. Description of symbols used in system model.

Symbol	Description	Symbol	Description
KW	Kilowatts used to measure Electricity	Inst	Instances
VM	Virtual Machine	BW	Band Width
P	Power consumption	exis	Existing
Core	CPU core	req	Requesting
MW	Middle Ware Tools	IAAS	Infrastructure As A Service
Console	Requested and needed Software packages	PAAS	Platform As A Service
VM_{image}	Kernel image depicted on VM	$Users_{add}$	Allotting resources to new users
W_{image}	Working Image(working Platform)	W_{image}	Working Image(working Platform)
VMCL	Virtual Machine Container List	PM	Physical Machine

7.1.1. Overloaded Host/ Destination Watcher

Heuristic approaches are developed to predict resource requirement to do execution [4], but it requires dynamic configuration and additional tools but our system made ease as follows, when computational resource demand of particular VM crosses the threshold level, then host id and user id are noted by overloaded watcher and started to extend the support of infrastructure service to save the execution without any outwits. If host is completely occupied and no more resources are available to create new VM'sthen universal searching of suitable VM with user ID to be performed.

7.1.2. Under Loaded Host/ Destination Watcher

During the task execution, when a particular PM engaged with less task then particular host Id, VM id are noted and forwarded to under loaded watcher. This help to carry the process of: it starts to shut down the possible instances by sharing among under loaded machines and over loaded applications are migrated to these instances.

7.2. Instance Consolidation Mechanism

Before engaging the task on particular VM, computational status of VM with threshold value is considered as key extinct to accept new tasks. Another consolidation mechanism is like; if two VM instances are running with the same environment and both are in under loaded threshold condition then automatic consolidation among the instance will be carried.

7.3. Resource Brokerage

This is very significant work in our proposed model, new VM's are created by considering the middleware tools and environmental platform from the user request. Then resource brokerage will serves the request by recording all the requirements with user id, host id, VM id and nature of application setup going to take place. And during execution any new supporting

resource required by customer, then request brokerage will arrange through PAAS.

8. The System Model

Initially, we start our model with Server Power consumption Minimization model, for that identification of general electrical consumption service was given by

$$Amps = (kW * 1000)/(volts * 1.73) \quad (1)$$

Overall power consumption for each server in data center are formulated by

$$P(s) = Con_{req} + \sum_{n=1}^{VM}(P_n VM) \quad (2)$$

But overall power consumed in each *VM* is determinedby number of CPU core utilized for a time

$$P(VM)=(CPUcore)\sum\left(\frac{Core}{UT}+\frac{Core}{ID}+\frac{Core}{SL},-\infty<x<\infty\right) \quad (3)$$

Virtual Machine Formation Model:
Converting *VM* as Instances based on the kernel image depicted Console MW

$$VM(Inst) = \sum_{instance=1}^{n} console_{MW} \ni \left(VM_{Image} \equiv VM_{Image}\right) \quad (4)$$
$$1-n$$

VM's are acting like an instance so middleware tool as well working images are instantly created instantly with the support of PAAS environment

$$VM(Inst) \approx PAAS_{sup} \forall VM(Inst) \ \forall IAAS_{sup} \quad (5)$$

$$r_{xy} = \sum_{i-1}^{n}(((x-\bar{x})(y-\bar{y}))/((x-\bar{x})(y-\bar{y}))^2)) \quad (6)$$

Equation (6) is obtained from Pearson correlation analysis. Based on this, user request and physical machines are correlated and task assigned.

$$U_{vm,}J,S=W_{image} \forall console_{MW} \forall CPU_{core} \subset IAAS_{sup} \quad (7)$$

$$U_{inst} = (CPU_{core} < BW) \stackrel{?}{=} exis(W_{image} * console_{MW}) \quad (8)$$

$$exis(W_{image}.console_{MW}) \not\equiv req(W_{image} * console_{MW}) \quad (9)$$

$$VM_{new}Inst=req(W_{image} * console_{MW}) \forall CPU_{core} \subset IAAS_{su|} \quad (10)$$

$$Users_{add}=\sum CPU_{core} > VM(inst) \forall (W_{image} * console_{MW}) \quad (11)$$

Instance measured with parameters like overloaded and under loaded by the terms of kernel image, Bandwidth, memory, Middleware tool. If resource availability is not able to continue the application then server consolidation or new VM instance will be created.

$$VM_{consol}=\sum CPU_{core} < M(inst) || CPU_{core} \asymp VM(inst \quad (12)$$

9. Algorithm Formulation

In this section, we briefly discuss the algorithms implemented in the mechanisms of 'Resource Brokerage', 'Host Status monitor' and 'Instance Consolidation' of proposed framework.

The algorithm first checks if the CPU work-load history of VM instances and hosts are adequate by using correlation analysis. In a case that the workload history is not available, it simply uses VM creation process. If the workload history is available, then perform threshold verification and accommodate with the VMs and update host id, user id on host status monitor. If no hosts are found, then new instance are created.

Algorathim 1: Resource Balancing Process

Input:$Res_{provisioner,}$
Output:VM creator &update VM container List.
for eachactive host(VM id, User id)
changein MW tools &Com_{req},
do(IAAS || PAAS)$_{extends}$ upto ($Host_{utilthreshold}$)
UpdateVMC List
end

VM Creation Process

Input: VM Container List
Output: Selected VM from VMCL
getU_{req}
while($U_{req} \equiv VM_{existing}$)
doAssign user ID ⇆(Host Id, VM Id)
Update VMCL
Elsealert → $Res_{provisioner}$
Update → VMCL(Host Id, VM id, User id)
newVM≡U_{req} is formed
thenassign user ID ⇆(Host Id, VM ID).
activateVM creator Modules
end

Overload / Under load Destination Selector

Input: over/ under loaded Host List, active host,
Output: Destination (host Id,VM id),
VM to migrate list
for eachactive host
SortCPU utilization
If(Resource Utilization>Threshold capacity)
follow VM Container List. Remove (VM)
Search Host←underloaded
assign new destination id →resource provisioner
Activate VM creator module
Update (VM id, User id) →VMC List.
end

The formulation and working of these algorithms will ensures an added innovative process on virtualization process i.e., performing VM process based on user request, without creating guest OS, correlated OS for correlating customers. In the same time any customer request new software module packages to support their application during mid of execution also, Platform As A Service (PAAS) make possible on this. Extracting the features on containers is nightmare. During computational demands, container goes suitable resource searching to carry the application, if capable resource not available means then new execution environment with all middleware should be created. This is not preferable for time initiative based businesses. In case of self-adaptive VM's, on top of Host OS Infrastructure As A Service (IAAS) is attached, which provide resource support without disturbing the application execution and splitting of

application based on available instance is possible which avoid new environments entries. All these process are governed by Host monitor and destination watcher, so all the intermediate communications and demand request are recorded and no chance for misleading between the systems occurrences. Next we see the experimental evaluation of above mention algorithms with the help ofcloudsim and Docker Cloud.

10. Experimental Setup

In this section, we validated the performance of containers as well as self-adaptive virtual machines. The entire setup was created on cloudsim as well on top of Docker cloud. The feasibility ratio like execution time, and VM utilized was measured by considering certain set of in-distinguished tasks assigned on self-adaptive VM and Docker enabled systems. From the parameters, number of VM or container utilized helps to predict resource utilization percentage and from execution time we can predict the performance capability of particular system and long-time of task completion leads to continuous running of servers, which results more carbon emission. The overall experimental setup was shown in Table 2, here we categorised task on basis of platform related matching jobs and non-matching jobs. We followed three different types of environment, in that different nature of task are assigned to measure the feasibility of self-adaptive virtual machines.

From the table, we got results shown in Figure 6 like number of VM assigned for matching and non-matching jobs with time taken to complete the task assigned.

11. Result Discussion

By referring Figure 5, we come to conclusion that, time taken by container to execute task assigned was two times greater than self-adaptive VM's.43 platform engines are in lag for just 100 population strength. From the utilization strategy it was resembled as If we increase the population count, surely containers will take more operating engines than self-adaptive VM. If complexities of system increases, which also increase unwanted burden on container results, slow down the utilization factor and increases server running time leads to heavy carbon emission.

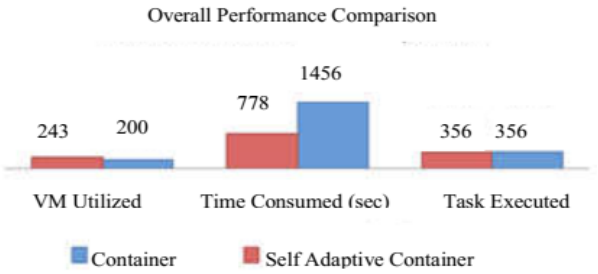
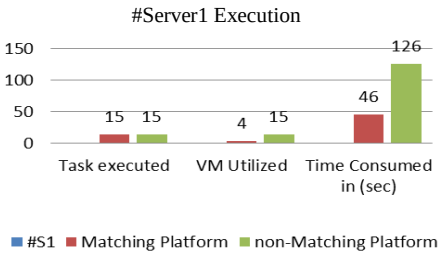


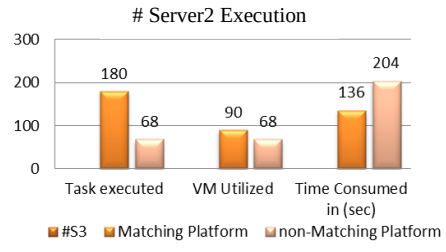
Figure 5. Performance comparison chart.

12. Conclusions

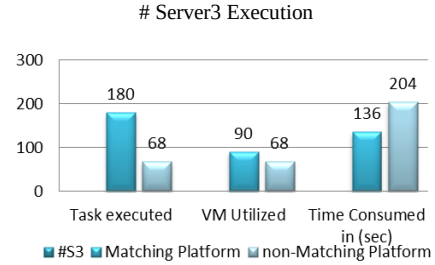
Our paper introduced new hybrid model to improve performance of VM's and proved progress better than containers. By increasing server utilization on execution environment, active server time ratio will automatically reduce and helps to tackle the issue of energy efficiency which also helps to reduce carbon footprints. Three set of simulations were carried and experimental results are pointing to improvement on performance of VM'sas well as utilization factor. This paper claim better way to improve utilization by: reducing overloaded VM's, rapid processing of assigned task and reduction of overall energy consumption by consolidating under loaded servers. In future we plan to go for improvements on VM placements in adaptive manner.



a) Server#1 response status.



b) Server#2 response status.



c) Server#3 response status.

Figure 6. Comparative end results of container and self adaptive container on various servers.

Table 2. Overall Experimental setup with sample task assigned.

Server Type	CPU[3GHZ] with 35000 MIPS	Memory (GB)	JOB Count	Task Assigned (similar and Dissimilar)	Types of job assigned*
#1	4 cores	64	5	30 (15&15)	SQL and Java based application, Web application; Hadoop based Jobs, Self-developed App on cloud Dockers, and Online-form Processing jobs.
#2	8 cores	128	15	78(20 & 58)	
#3	16 cores	256	40	248(180&68)	

References

[1] Abdelbaky M., Diaz-Montes J., Parashar M., Unuvar M., and Steinder M., "Docker Containers Across Multiple Clouds and Data Canters," *in Proceedings of IEEE/ACM 8th International Conference on Utility and Cloud Computing*, Limassol, pp. 368-371, 2015.

[2] Celesti A., Mulfari D., Fazio M., Villari M., and Puliafito A., "Exploringcontainer Virtualization in IoT Clouds," *in Proceedings of IEEE International Conference on Smart Computing*, MO, pp. 1-6, 2016.

[3] Dhakate S. and Anand G., "Distributed Cloud Monitoring Using Docker As Next Generation Container Virtualization Technology," *in Proceedings of Annual IEEE India Conference*, New Delhi, pp. 1-5, 2015.

[4] Gao K., Wang Q., and Xi L., "Reduct Algorithm Based Execution Times Prediction in Knowledge Discovery Cloud Computing Environment," *The International Arab Journal of Information Technology*, vol. 11, no. 3, pp. 268-275, 2014.

[5] Gerlach W., Tang W., Keegan K., Harrison T., Wilke A., Bischof J., D'Souza M., Devoid S., Murphy-Olson D., Desai N., and Meyer F., "Skyport:Container-Based Execution Environment Management for Multi-Cloud Scientific Workflows," *in Proceedings of 5th International Workshop on Data-Intensive Computing in the Clouds*, New Orleans, pp. 25-32, 2014.

[6] He S., Guo L., Guo Y., Wu C., Ghanem M., and Han R., "Elastic Application Container: A Lightweight Approach For Cloud Resource Provisioning," *in Proceedings of 26th International Conference on Advanced Information Networking and Applications*, Fukuoka, pp. 15-22, 2012.

[7] Huang W. and Knottenbelt W., "Self Adaptive Containers: Interoperability Extensions and Cloud Integration," *in Proceedings of 14th International Conference on Scalable Computing and Communications and its Associated Workshops*, Bali, pp. 433-440, 2014.

[8] Internet Usage and Social Media Statistics, www.internetlivestats.com, Last Visited, 2016.

[9] Kamarainen T., Shan Y., Siekkinen M., and Yla-Jaaski A., "Virtual Machines vs. Containers in Cloud Gaming Systems," *in Proceedings of International Workshop on Network and Systems Support for Games*, Zagreb, pp. 1-6, 2015.

[10] Liu K., Aida K., Yokoyama S., and Masatani Y., "Flexible Container-Based Computing Platform on Cloud for Scientific Workflows," *in Proceedings of International Conference on Cloud Computing Research and Innovations*, Singapore, pp. 56-63, 2016.

[11] Mohamed M., Belaid D., and Tata S., "Self-Managed Micro-containers for Service-Based Applications in Cloud," *in Proceedings of Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Hammamet, pp. 140-145, 2013.

[12] Shanmugam S. and Iyengar N., "Effort of Load Balancer to Achieve Green Cloud Computing: A Review," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 11, no. 3, pp. 317-332, 2015.

[13] Varghese B., Subba L., Thai L., and Barker A., "Container-Based Cloud Virtual Machine Benchmarking," *in Proceedings of IEEE International Conference on Cloud Engineering*, Berlin, pp. 192-201, 2016.

[14] Xu X., Yu H., and Pei X., "A Novel Resource Scheduling Approach in Container Based Clouds," *in Proceedings of IEEE 17th International Conference on Computational Science and Engineering*, Chengdu, pp. 257-264, 2014.

[15] Zhang J., Lu X., and Panda D., "High Performance MPI Library for Container-Based HPC Cloud on Infini Band Clusters," *in Proceedings of 45th International Conference on Parallel Processing*, Philadelphia, pp. 268-277, 2016.

Siva Shanmugam (born 1987), working as an Asst.Prof.& Research Scholar at SCOPE, VIT University, Vellore, TN, India. He has had 6 years of teaching experience and currently doing his PhD research on Green Cloud computing.His interest includes Distributed Systems, Web Security, Sensor Networks, Mobile and Internet Computing.

Sriman Iyengar (born 1961) is currently Senior Professor at the School of ComputerScience &Eng. VITUniversity, Vellore-632014, TN,India. His interests include Intelligent Computing, Networks and Security applications &Cloud Computing. He authored several textbooks and published nearly 250 research articles in Scopus indexed journals. He served as Keynote Speaker/Invited/plenary Speaker for many International conferences. He handled different projects along with his students.He is member of CSI, ISTE, ACM and many.