International Journal of Electrical and Computer Engineering (IJECE)
Vol. 8, No. 3, June 2018, pp. 1659~1670
ISSN: 2088-8708, DOI: 10.11591/ijece.v8i3.pp1659-1670

CCCORE: Cloud Container for Collaborative Research

Salini Suresh[1], L. Manjunatha Rao[2]
[1]Computer Science, Bharathiar University, Coimbatore, India
[2]Department of MCA, Dr. Ambedkar Institute of Technology, Bangalore, India

Article Info

Article history:

Received Jun 9, 2017
Revised Jan 1, 2018
Accepted Jan 8, 2018

Keyword:

Cloud computing
Collaborative research
Container
Dynamic allocation
Finish time
Residual resources
Throughput

ABSTRACT

Cloud-based research collaboration platforms render scalable, secure and inventive environments that enabled academic and scientific researchers to share research data, applications and provide access to high- performance computing resources. Dynamic allocation of resources according to the unpredictable needs of applications used by researchers is a key challenge in collaborative research environments. We propose the design of Cloud Container based Collaborative Research (CCCORE) framework to address dynamic resource provisioning according to the variable workload of compute and data-intensive applications or analysis tools used by researchers. Our proposed approach relies on–demand, customized containerization and comprehensive assessment of resource requirements to achieve optimal resource allocation in a dynamic collaborative research environment. We propose algorithms for dynamic resource allocation problem in a collaborative research environment, which aim to minimize finish time, improve throughput and achieve optimal resource utilization by employing the underutilized residual resources.

Corresponding Author:

Salini Suresh,
Computer Science,
Bharathiar University,
Coimbatore, India.
Email: pnsalinisuresh@gmail.com

1. INTRODUCTION

In mid-1990's various grid-based cyberinfrastructures or e-infrastructures were constituted that integrated high-speed research networks and middleware services and endorsed researchers for collaborative sharing of distributed resources. These firmly unified science gateways served as resource providers for specialized as well as generic research initiatives [1]. However, restricted interface to the data, domain-specific nature of science gateways did not match the requirement of the researchers outside those domains [2]. With the advent of cloud computing, easy reconfigurable and adaptive Virtual private research environments and science clouds became a preferred alternative to a traditional grid or cluster-based e-infrastructures. Cloud-based collaborative research platforms provide the researchers with computing, storage resources required to run their applications, and they can collaborate to share data and application, while he concentrates on his area of research. Cloud platform offers compute environment with the huge set of computing resources much bigger than what an individual research organization can afford. Organizations can scale up, scale down the resources, and pay for it according to the usage. Multitenancy provided by cloud architecture enabled the creation of domain and requirement specific virtual private research environments that expedited researchers for collaboration and sharing of the resources [3]. Several science clouds such as Nectar Research cloud [4] provides the infrastructure to run compute-intensive scientific applications [5], [6]. Even though a substantial amount of research work has been carried out with regard to cloud-based collaborative research platforms, ample work does not exist in view of dynamic resource allocation in collaborative research cloud frameworks.

The primary aim of the paper is to design a Cloud Container based Collaborative Research (CCCORE) framework employing an on-demand, dynamic resource provisioning according to the varying workload, through a comprehensive assessment of requirements of the users and available resources in a collaborative research environment.

1.1. Background

In this section, we discourse an illustrative set of existing work related to cloud-based collaborative research platforms, among which some platforms used hypervisor-based virtualization while others have deployed containerization based resource allocation.

Benjamin H. Brinkman et al [7] proposed a cloud-based portal for sharing data and collaborating on projects containing large EEG datasets for fostering collaborative research. Authors discuss that portal provides fundamental requirements of collaborative research platform and some of the features they have emphasized are the security of the data and access rights on the data, access to data and results of an analysis, a platform independent tool to view and search datasets.

Tarek Sherif et al [8] proposes a CBRAIN, a web-based generic collaborative research platform that offers access to remote data sources, distributed computing sites, processing and visualization tools for data and compute-intensive research in neuroimaging.

A. Mc Gregor et al [9] present RP-SMARF, a collaborative research platform built on cloud, in the area of smart facilities management, which connects geographically disseminated heterogeneous resources.

Bastian Roth et al [10] have sort after the challenges in scientific collaboration and proposed an approach, which leverages on groupware tools and hypervisor-based virtualization techniques like KVM, VMware vSphere or Xen to run a generic collaboration platform.

Muhamad Fitra Kacamarga et al [11] authors put forward complete computing platform in bioinformatics research, which uses Docker containers for lightweight virtualization. Paper describes that Docker containers allow customization of the compute environment and effectively overcome the challenges in VM based approach.

Yujian Zhu et al [12] demonstrates a lightweight container based and a scalable system called Docket is based on LXC (Linux Containers) which provides a platform to run different application frameworks pertaining to academic and scientific research.

Elahehkheiri et al [13] have elaborated a tenant-based resource allocation approach using genetic algorithm and heuristic algorithm to overcome the issues of over-utilization and under-utilization in resource allocation for SaaS applications.

Sijin He et al [14] have proposed a virtual resource unit named EAC, which delivers better resource efficiency and scalability and discussed resource-inefficiency in the VM-based approach.

1.2. Problem

Scientific research in various disciplines often involves researchers from different organizations collaborating to conduct analysis, experiments or simulations that are data and compute intensive and with unpredictable resource requirements [15], [16]. These kind applications or tools requires highly dynamic resource allocation method. The resource intensive applications, data, and tools shared in highly collaborative research platforms suffer from bursty workloads [17]. However, most of the collaborative research platforms depend on the Cloud service providers for resource provisioning that schedule the applications independently and provisions the resources statically. Lack of a comprehensive assessment of applications and the available resources can lead to under or over utilization of resources and increased execution time for an application [18], which is undesirable in a collaborative research environment.

Therefore, we identified that the major problems as for resource allocation in collaborative research cloud frameworks with varying workloads are:

a. Bursty workloads owing to Data and compute-intensive tools and applications.
b. Static provisioning of resources, which leads to resource locking.
c. Increased execution time due to lack of comprehensive assessment of applications and the available resources.

1.3. Proposed solution

Our proposal is the design of Cloud Container based Collaborative Research (CCCORE) framework that intends on– demand, customized containerization, comprehensive assessment of resource requirements and applies a scalable algorithm that uses underutilized residual resources to achieve optimal resource allocation in a dynamic collaborative research environment. CCCORE offers a proficient way to standardize research methods, establish a relationship among data, and share the findings amongst researchers.This enables the researcher to focus on his domain of research rather than gaining the proficiency in infrastructure

installations and analysis tools [19]. CCCORE rapidly spawns computational instances and provide a customized unit of resources according to the varying workload of applications or tools used by the researcher [20]. Researchers often need to replicate the results, study the inferences or analyze the results by varying the parameters. CCCORE containerizes entire set of data, application and all its dependencies, hence deliver a complete compute environment for the researcher.

2. ARCHITECHTURE OF CCCORE

2.1. CCCORE components

CCCORE integrates two units a) Research collaboration unit (RCU) and b) Management Interface (MI). RCU is ready to use container with data, applications/ tools, and operating system. It is optimized based on a finish time. RCU is shared among collaborating researchers on a trusted network. The residual resource pool of RCU provides it the capability to run an instance of an application and create an operating image for theresearcher. Figure 1 demonstrates the model of an RCU.

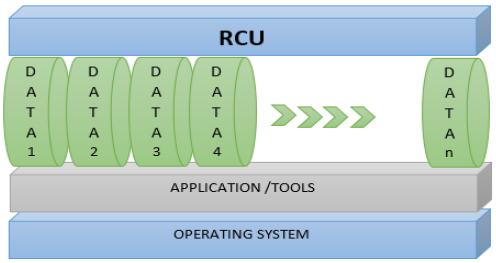

Figure 1. Model of RCU

We defined the original researcher who owns the research data, application or tools as owner. MI manages and administers RCU. Researcher sends the login request to the owner through MI. Owner approves or denies the login request depending on the credentials. When researcher request for the resources, MI will check resources available with owner and provision RCU from his pool of resources. The CCCORE defines permissions to view, edit, delete and publish the data and applications in the container based on user rights. The owner through MI set researcher's rights on RCU through Access control list (ACL). The two conditions that arise in setting the rights of the researcher are:

a. The owner gives the researcher full rights on RCU and owner rolls backs his rights on it.
b. Owner and researcher collaborate and hold the same rights on RCU.

Table 1. Researcher's Rights on RCU

Rights	Description
No Access	The Researcher will not see the RCU in his account.
View	The Researcher can see the RCU in his account and can view the data and tools available in the RCU.
View and execute	The Researcher can view the data and work on the data with tools available in a different parameter setting.
Ownership	Researcher will own RCU.

2.2. Sequence diagram of CCCORE

The stepwise description of the sequence diagram is given below:

Step 1: Researcher request for resources to MI
Step 2: MI verifies researcher and authenticate.
Step 3: MI sends query research request to owner.
Step 4: Owner verifies the request, authenticate and allocate resources packaged in RCU.
Step 5: RegisterRCU details (allocated memory, CPU, storage, bandwidth) with MI.
Step 6: Set researcher rights on RCU and grant it to researcher.
Step 7: Researcher access RCU.
Step 8: MI monitors RCU performance for under provisioning or over provisioning.
Step 9: MI manages RCU the resource and resource allocation.
Step 10: MI optimizes RCUfor better finish time and resource utilization.

Step 11: Researcher sends the decommission request to MI upon finishing the job.
Step 12: MI decommissions RCU by releasing the resources.
Step 13: MI update the resource pool of RCU.
Step 14: MI update the RCU decommission to owner.

Figure 2 shows the sequence Diagram of CCCORE

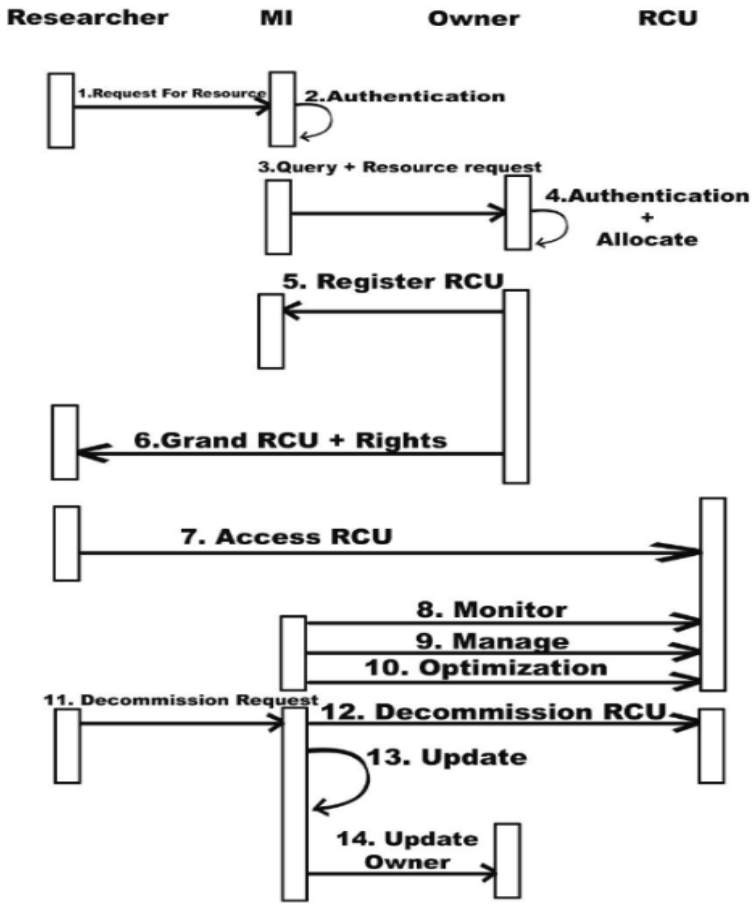

Figure 2. Sequence Diagram of CCCORE

2.3. CCCORE capabilities

In the following section, we describe some of the key potentialities of CCCORE as a collaborative research platform.

Customization: CCCORE creates custom-built RCUs on demand according to researcher's requirements. A researcher can select data (raw or analyzed), applications, and compute, storage resources bundled as RCU.

Flexibility: Inthe scientific research analysis, researcher may often need to build multiple environments, to generate various results based on the parameter settings. CCCORE enables the researchers to work on an existing project by duplicating the same settings irrespective of the local host environment [21]. CCCORE setsan environment to run multiple instances of same applications for different users.

Reproducibility: Reproducibility of researchis time consuming and challenging and call for configuring the platform, virtual machine clustering, compatibility fixes for operating system, software libraries andtools [22]. CCORE expounds reproducibility to facilitate researchers to reproduce the complete compute environment used by the original researcher. CCCORE create lightweight RCUswith an entireset of data, application and all its dependencies like root file systems, registries, software libraries and thus the entire workflow of a project used by the original researcher could be replicated and extended byother

researchers. Research findings and inferences packaged in RCU is shared and reused by other researchers, thus facilitating validationof theresults and inferences.

Computational portability: Some computational tools used for scientific analysis tightly couples with system environments and registry settings. RCU being a lightweight container and platform independent is portable across all platforms. The replication of the computational environments to run the applications shared between researchersis resolved in CCCORE as RCU instances can be exported to any environment, consequently enabling the emulation of computational environments to run these applications. Open Virtualization Format (OVF) defines an open source standard for packaging and distributing software for virtual machines.

Dynamic resource provisioning: CCCORE count on autoscaling tofurther dynamic allocation of resources for compute intensive research applications. Research tools or applications may demand set of dedicated resource or at times workload can vary based on the intensity of analysis. Scalability [23] imparted in CCCORE enables allocation of resourcesin response to the uncertain workload.Demand-driven resource provisioning commissions or decommissions resource instances for the RCU through MI.To achievea faster execution time, MI allocates residual resources of any RCU to any other RCU that demands it. Provisioning the compute capacity according to the varying workload that occurs in scientific applications requires the elimination of resource locking due to static provisioning of resources. Moreover, the static resource provision causes under utilization or over utilization of resources that poses a challenge in resource allocation.

2.4. Framework of CCCORE

The main modules of the layered framework of CCCORE are Physical layer, virtualization and control layer, service layer, delivery layer. Figure 3 illustrates layered framework architecture of CCCORE.

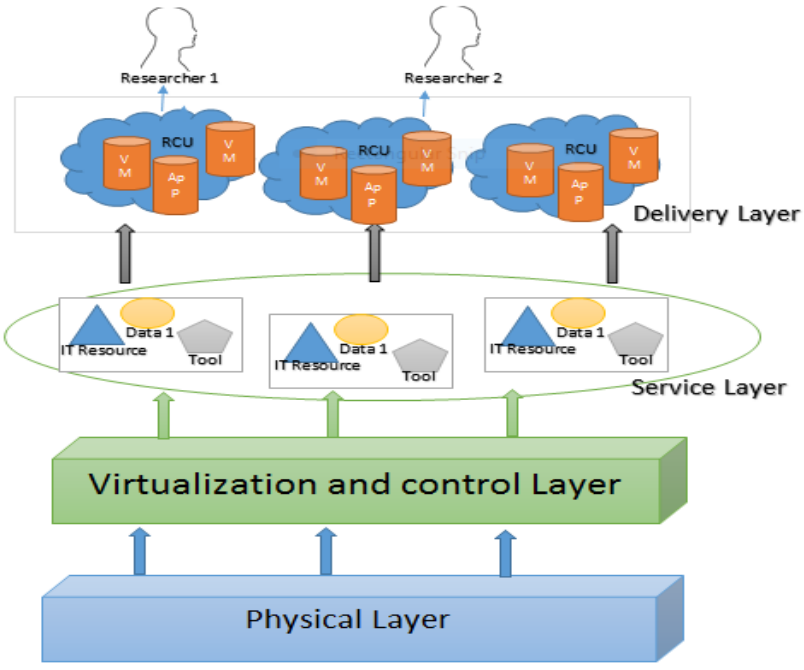

Figure 3. Layered architecture of CCCORE

Physical layer: Physical layer allocates necessary compute, storage and bandwidth to create the compute stack of any RCU. Two virtual routers interconnect multiple virtual resources. MI creates RCU of different configurations according to the researcher's needs. MI specifies the virtual path depending on the bandwidth allocated to each researcher. Virtual infrastructure Diagram of CCCORE in Figure 4 illustrates the interconnection of virtual resources of CCCORE.

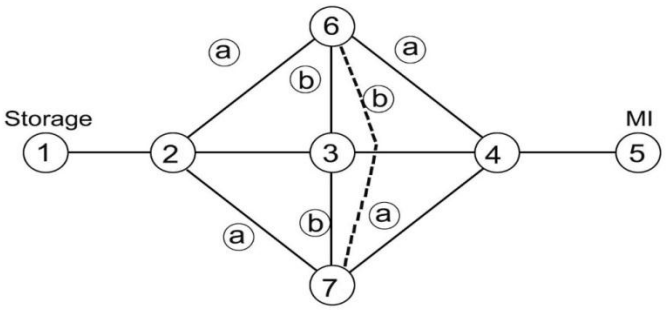

Figure 4. Virtual infrastructure diagram of CCCORE

Table 2 shows the functionalities of each node of Virtual infrastructure diagram.

Table 2. Node Functionalities

NODE Number	FUNCTION
1	Storage size (hard disk size)
2 ,4	Virtual Router
3,6,7	Compute nodes
5	MI

We consider *a* as the virtual link bandwidth between virtual router and computational resources *b* as thevirtual limit latency between compute nodes. MI connects the resources (storage, compute) through virtual routers. To create an RCU, MI selects one of the computation nodes 3, 6, 7 based on the workload, through virtual router 4 creating routes 5-4-6, 5-4-7 or 5-4-3. MI connects Computation nodes 3, 6, 7 to storage node 1 through virtual router 2. MI comprehensively assess the available resources of CCCORE and allocates bandwidth and resourcesto any RCU based on workload requirement and finish time.

Virtualization and control layer: In Hypervisor based virtualization; the guest operating system that runs the applications consumes server resources thus increasing the system overheads [24]. Virtualization and control layer has adopted operating system level virtualization that enables the RCUs to share the operating system with host and other RCUs [25]. The layer offers an abstraction for the researchers and ensures isolation of resources for all the RCUs.

Service layer: This layer acts as a repository, which storesimages inOVF (Open Virtual Format) of all RCUs .RCU is exported in OVF format to the image depo. OVF format enhances theportability and platform independenceof RCU. Researchers access the allocated RCUthrough the service layer.

Delivery layer: In a collaborative research environment where resource demands are always high, Virtual Machine (VM) based approach can be in efficient. Delivery layer counts on rapidly scalable containers to accommodate high resource demands [26].

3. RESEARCH METHODOLGY

3.1. System model

We model dynamic resource allocation problem as an optimization problemand aims to minimize the finish time and improve the throughput to achieve optimal resource utilization. Our container based resource allocation algorithm enhances dynamic scalability by employing underutilized residual resources [27] and hence minimize finish time of an application.

Consider the set of total available resources N^p (compute, memory, storage, and bandwidth) in CCCORE. Each RCU is denoted as r, residual resources in each RCU is denoted b_r^p. Consider job (application) Aj with workload L_j, and maximum allowed service delay T_j, then the resources required R_j^u is calculated as

$$R_j^u = \frac{L_j}{T_j} \quad (1)$$

RCU will not execute a job with a size less than defined minimum value to avoid under utilization and resource locking. We define a minimum size of any job executed by RCU.

Minimum job size shouldbe ≥ $L_j\beta_j$where $\beta_j = \frac{\text{smallest workload}}{Lj}$

MI comprehensively assess the total available resources in CCCOREto optimally allocate resources.
Total residual resources in RCU is calculated as,

$$Z = \sum_r^{N^p} b_r^p \quad (2)$$

Finish time for a job is a ratio of workload to resource required with a specific time delay. Finish time decreases with optimal utilization of residual resources.

Finish time for a Job A_j is calculated as,

$$F_j^T = \frac{A_j}{R_j^u} - \frac{A_j}{(\frac{Z}{T_j})} \quad (3)$$

Let Z_r is allocated bandwidth for each user, Y is the unused bandwidth for RCU, n is the maximum number of RCUs that can be created in CCCORE, x is active RCUs at any moment of time.
Maximum throughput allocated to any RCU (X_r) is calculated as:

$$X_r = Z_r + \sum_{n-r}^{n} Y \quad (4)$$

Maximum throughput of CCCORE is calculated as $\sum_1^x a - \sum_1^{x-1} b$

3.2. Proposed algorithm

An on–demand, flexible resource provisioning call for a comprehensive assessment of requirements of the users and available resources. The proposed algorithm aims to minimize the finish time, improve the throughput and achieve optimal resource utilization. If the initially provisioned resources of an RCU is not adequate either to meet the finish time or resource requirements of an application, MI allocate the requested resources from the unused residual resources of other RCUs.

Algorithm 1: RCU Allocation
Input:
A: Maximum number of RCUs allocated for each researcher owner
N: Total number for RCUs available in CCCORE
B: Maximum number RCUs any researcher can request.
Output: RCU $_{ij}$
1. If B ≤ A then
2. Obtain RCU $_{ij}$(1≤B ≤A) from A
3. Create RCU $_{ij}$
4. If B> A then
5. Obtain RCU $_{ij}$(A ≤ B ≤ N) from N with MI approval
6. Create RCU $_{ij}$
7. Set user rights for RCU $_{ij}$

Algorithm 2: Optimal resource allocation algorithm
Input:
N^P: Total Available resources in CCCORE.
r: RCU number
Job: A_j
Work Load: L_j
Max allowed time delay: T_j
Minimal resource required for job $A_j R_j^u = \frac{L_j}{T_j}$
Residual resource in RCU $Z = \sum_a^{N^p} b_a^p$
a = bandwidth of RCU

b =Latency between RCUs
Output: F_j^t, optimizedresource
1. Job /workload requested
2, Created RCU r
3. Resource allocated to RCU r $\leftarrow R_j^u$ $(R_j^u > M_j^p)$
4. Finish Time $F_j^T = Ai \div R_j^u$
5. If actual finish time >Tj
6. Residual resource added to RCU r $\leftarrow (R_j^u + Z)$
7. Finish time = $F_j^T = \frac{A_j}{R_j^u} - \frac{A_j}{(\frac{Z}{T_j})}$
8. If required resource is more than R_j^u
9. Resource added to RCU r $\leftarrow (R_j^u + Z)$
10. Finish time = $F_j^T = \frac{A_j}{R_j^u} - \frac{A_j}{(\frac{Z}{T_j})}$
11. If idle time of RCU>I
12. RCUdecommissioned.

4. RESULT AND ANALYSIS

The hardware infrastructure deployed for the experiment consisted is as follows:
Identical configuration of four physical machines each with configuration core i5 5287U processor 3 MB smart cache, 2 core /4 threads @ 2.9 GHz. Installed Memory (RAM): 4.00 GB which are connected using 1G Ethernet switch cisco SF 300 -24 port.We configured RCU based systemswith Physical machines installedwith Ubuntu 14.04, Open stack and 4 LXD (Linux containers).VM based systems are installed with windows 2012 server Standard edition with service pack 2 and 4 VMs.

4.1. Scenario I

We evaluated VM-based and the RCU-based systems for resource efficiency with respect to finish time and throughput. Improvement of finish time, increases the resource efficiency in a collaborative research environment. We compared the VM-based and RCU-based systemsby running a .net application and Sage Math. While the .net application is computationally light, sage math is a memory and compute intensive application. We conducted multiple iterations by varying configuration of VM and RCU. We conducted 50 iterations for .net application, as it is lightweight and 10 iterations for Sage math.Table 3 showsaverage finish time in executing the .net application for configurations 1) 2core compute, 1GB RAM and 2) 4core compute, 8GB RAM and in executing Sage Math application for configuration s3) 6core compute, 8GB RAM and 4) 8core compute, 16GB RAM using VM based and RCU based systems.

Table 3. Average finish time for VM and RCU using .net Application and Sage Math

Configuration	Application	Iterations	Type	Average finishtime in seconds
2core,1GBmemory	.net	50	VM	83.06
2core,1GB memory	.net	50	RCU	45.18
4core,8GB memory	.net	50	VM	82.7
4core,8GB memory	.net	50	RCU	38.14
6core,8GB memory	SageMath	10	VM	1839.56
6core,8GB memory	SageMath	10	RCU	1086.33
8core,16GB memory	SageMath	10	VM	1839
8core,16GB memory	SageMath	10	RCU	952

Figure 5 highlights that RCU showed 45% better finish time than VM for configuration 1) 53.8% better finish time for configuration, 2), 41% better finish time for configuration, 3) 48% better finish time for configuration, 4) the comparative analysis highlights that with increase of resources (core and memory) our proposed RCU based CCCORE delivers a better finish timethan VM, due to improved resource utilization implemented through our algorithm.

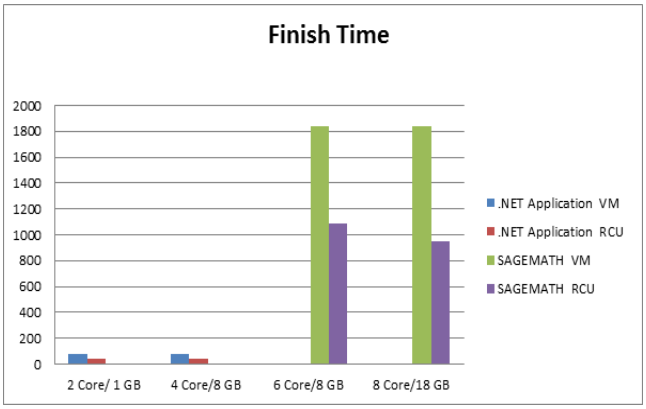

Figure 5. Comparative analysis of Average finish time for VM and RCU

4.2. Scenario II

To evaluate the dynamic allocation of resources in line with the workload of compute-intensive applications, wecalled functions for Bernoulli number, Integer factorization, and factorial in SAGE Math. Bernoulli number function is computeand memory intensive whereas Integer factorization and factorial functions are less compute intensive.

We compared finish time of VM, LXD and RCU systems withan identical configuration of 8 core, 16GB RAM in three iterations varying the residual resources. By varying the residual resources, we analyzed the impact of resource optimization in the finish time. In the first iteration, no residual resources were made available in the system; second iteration, with 25% residual resources available, in the third iteration 60% residual resources were available.

Table 4. Finish time for VM, LXD, and RCU using Compute Intensive Sage Math Functions

APPLICATION	VM Finish time in Sec	LXD Finish time in Sec	RCU Finish time in Sec
BERNOULI NUMBER			
no residual resource	248	221	221
25% residual resource	248	221	177
60% residual resource	248	221	160
INTEGER FACTORISATION			
no residual resource	170	155	150
25% residual resource	170	155	134
60% residual resource	170	155	113
FACTORIAL			
no residual resource	39	32	32
25% residual resource	39	32	24
60% residual resource	39	32	13

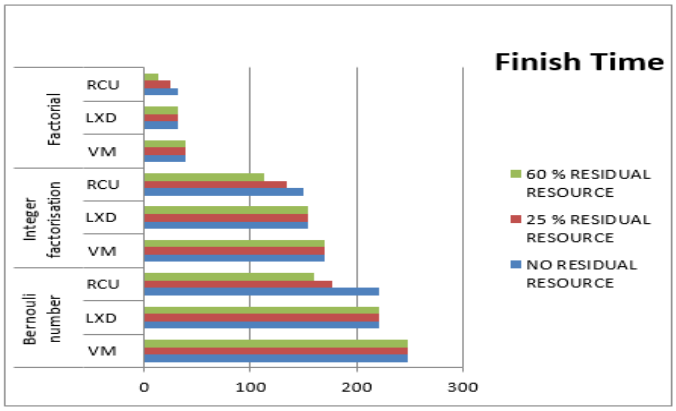

Figure 6. Comparative analysis of finish time for VM, LXDRCU with available residual resources

The comparative analysis shown in figure 6 demonstrates that finish time for VM and LXD did not change with the availability of residual resources, but RCU employed underutilized residual resources and achieved a better finish time.

4.3. Scenario III

We conducted experimentsto evaluate the throughput of RCU and VM for processing data in varying sizes (1 GB, 4GB, and Bulk data ≥100 GB). The purpose of thestudy is to analyse the efficiency of RCU inutilizing the unused bandwidth toachieve better throughput as shown in Table 5.

Table 5. Comparison of Throughput for VM and RCU

Configuration	Data	Iterations	TYPE	Throughput in Gbps
2core, 4GBRAM 500GB harddisk.	1GB	10	VM	149.4
2core, 4GBRAM 500GB harddisk.	1GB	10	RCU	169.2
2core,4GB RAM,500GB harddisk	4GB	10	VM	140.5
2core,4GB RAM,500GB harddisk	4GB	10	RCU	174.7
8core,16GB RAM,500 GB hard disk	Bulk data≥100 GB	10	VM	139.3
8core,16GB RAM,500 GB hard disk	Bulk data≥100 GB	10	RCU	181.3

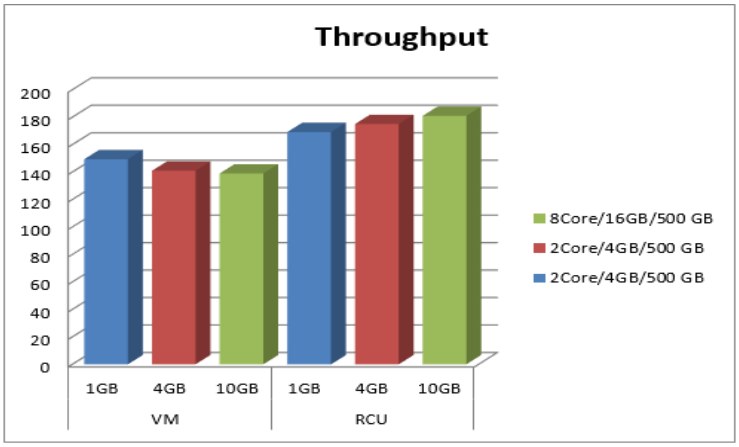

Figure 7. Comparison of throughput of VM and RCU in processing data of varying sizes

As it is obvious from Figure 7, while migrating 1GB data, RCU systems deliver improved throughput of 13% more than the throughput of VM based systems. Throughput increased by 24% with data of 4GB and 30.15% with bulk data migration. Therefore RCU achieves a better throughput compared to VM in processing data in variable sizes since is able to use the unusedbandwidth to achieve better throughput.

5. CONCLUSION

We have designeda Cloud Container based Collaborative Research (CCCORE) framework with dynamic resource provisioning according to the varying workload in the collaborative research environment. The proposed system relies on flexible, customized containers named as RCU to spawn complete computational environment for the researchers. Comprehensive assessment of user's requirements and using underutilized residual resources enhanced the efficiency of CCCORE. Experimental evaluation indicates that proposed RCU based CCCORE framework outperformed VM based systems in terms of finish time and throughput. Our future work will comprise the workflow automation of CCCORE and improve the container security.

REFERENCES

[1] Kalyanam, *et al*., "Cloud-enabling a Collaborative Research Platform: The GABBs Story", *PEARC17 in Proceedings of the Practice and Experience in Advanced Research Computing*, no. 23, 2017.

[2] Marty Humphrey, *et al*., "CloudDRN: A Lightweight, End-to end System for Sharing Distributed Research Data in the Cloud", In *Proceedings of IEEE 9th International Conference on e-Science*, pp. 254-26, 2013.
[3] Voss, *et al*, "An elastic virtual infrastructure for research applications (ELVIRA)", *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 2, no. 20, 2013.
[4] Sehrish Kanwal, *et al*, "Challenges of Large-Scale Biomedical Workflows on the Cloud -- A Case Study on the Need for Reproducibility of Results, In *Proceedings of IEEE 28th International Symposium on Computer-Based Medical Systems*, pp. 220-225, 2015.
[5] K. Keahey, *et al*, "Science clouds: Early experiences in cloud computing for scientific applications", *in Proceedings of Cloud Computing and Application Workshop*, pp. 825-830, 2008.
[6] Surya Nepal, *et al*, "TruXy: Trusted Storage Cloud for Scientific Workflows", *IEEE transactions on cloud computing*, vol. 5, no. 3, pp. 428-441, 2017.
[7] Benjamin H. Brinkmann, *et al*, "A Multimodal Platform for Cloud-based Collaborative Research", *In Proceedings of IEEE EMBS Conference on Neural Engineering*, pp. 1386-1389, 2013.
[8] Tarek Sherif, *et al*, "CBRAIN: A web based, distributed computing platform for collaborative neuroimaging research", *Frontiers in Neuro informatics*, vol. 8, article 54, pp. 1-13, 2014.
[9] A. McGregor, *et al*, "A Cloud-Based Platform for Supporting Research Collaboration", In *Proceedings of IEEE 8th International Conference on Cloud Computing*, pp 1107-1110, 2015.
[10] Bastian Roth, *et al*, "Towards a Generic Cloud-based Virtual Research Environment", *Computer Software and Applications Conference Workshops (COMPSACW)*, pp. 267-272, 2011.
[11] Muhamad Fitra Kacamarga, *et al*, "Lightweight Virtualization in Cloud Computing for Research", *Communications in Computer and Information Science*, vol. 516 Springer International Publishing, pp. 439-445, 2015.
[12] Yujian Zhu, *et al*, "Monitoring and Billing of A Lightweight Cloud System Based on Linux Container", *InProceedings of International Conference on Distributed Computing Systems Workshops IEEE*, pp. 325-329, 2017.
[13] Elahehkheiri, *et al*, "An Approach based on Genetic Algorithm or multi-tenant Resource Allocation in SaaS Applications", *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 6, no. 3, pp. 124-138, 2017
[14] Sijin He, *et al*, "Elastic Application Container: A Lightweight Approach for Cloud Resource Provisioning", *In Proceedings of IEEEInternational Conferenceon Advanced Information Networking and Applications (AINA)*, pp. 15-22, 2012**.**
[15] Xiaoquan Su, *et al*, "An Open – source Collaboration Environment for Metagenomics Research", *In Proceedings of IEEE International Conference on e-Science*, pp. 7-14, 2011
[16] Yongzheng Ma, *et al*, "Scientific collaboration cloud platform and its multidisciplinary applications", *In Proceedings of IEEE Conference Information Science and Technology (ICIST)*, pp. 466-470, 2013
[17] A. Rosenthal, *et al*, "Cloud computing: A new business paradigm for biomedical information sharing", *Journal of Biomedical Informatics*, vol. 43, no. 2, pp. 342-353, 2010.
[18] Uchechukwu Awada, *et al*, "Improving Resource Efficiency of Container-instance Clusters on Clouds", *In Proceedings ofIEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 929-934, 2017.
[19] J.J. Rehr, *et al*, "Scientific Computing in the Cloud", *Computing in Science & Engineering*, vol. 12, no. 3, pp. 34-43, 2010.
[20] Yan Hu, *et al*, "A *cloud computing solution for sharing healthcare information". In Proceedings of the 7th International Conference for Internet Technology and Secured Transactions (ICITST)*, pp. 465-470, 2012.
[21] Carl Boettiger, "An introduction to Docker for reproducible research, with examples from the R environment", *ACM SIGOPS Operating Systems Review - Special Issue on Repeatability and Sharing of Experimental Artifacts*, vol. 49, no. 1, pp. 71-79, 2015.
[22] Svetlana Sveshnikova, *et al*, "Using Virtualisation for Reproducible Research and Code Portability", *In Proceedings of International Conference on High Performance Computing & Simulation (HPCS)*, pp. 891-89, 2017.
[23] BaharAsgari, *et al*, "An Effiecient Approach for Resource Auto-Scaling in Cloud Environments", *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 6, no. 5, pp. 2415-2424, 2016.
[24] Jyoti Shetty, *et al*, "An Empirical Performance Evaluation of Docker Container, Openstack Virtual Machine and Bare Metal Server", *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 7, no. 1, pp. 205-213, 2017
[25] David Beserra, *et al*, "Comparing the performance of OS-level virtualization tools in SoC-basedsystems: The case of I/O-bound applications", *In Proceedings of IEEE Symposium on Computers and Communications (ISCC)*, pp. 627-632, 2017.
[26] Luciano Baresi, *et al*, "*MicroCloud*: A Container-based Solution for Efficient Resource Management in the Cloud", *In Proceedings of IEEE International Conference on Smart Cloud*, pp. 218-223, 2016.
[27] Bruno Yuji Lino Kimura, *et al*, "Workload regression-based resource provisioning for small cloudproviders", *In Proceedings of IEEE Symposium on Computers and Communication (ISCC)*, pp. 295-301, 2016.

BIOGRAPHIES OF AUTHORS

Salini Suresh is working as Assistant Professor, Department of Computer Science, Seshadripuram Academy of Business Studies, Bangalore. She has got 12 years of teaching experience. She has obtained Bachelor of Science from Calicut University in the year 1997, Master of Computer Application from Bharathidasan University in the year 2000 and Master of Philosophy from Manonmaniam Sundurnar University in the year 2003. Now she is a Research scholar at Bharathiar University, Coimbatore, India. She has authored 3 textbooks and published research papers in one International Journal and presented papers at National and International conferences.

Dr. L. Manjunatha Rao is working as Professor and Head, Department of MCA, Dr. AIT, Bangalore. He has 25 years of teaching experience. He did his Bachelor of Science from Bangalore University in the year 1990. He Studied Masters of Computer Application from Madurai Kamaraj University and was awarded in the year 1999. In 2002 did Master of Philosophy from Manonmanium Sundaranar University. He has awarded Ph.D. from Vinayaka Mission University, Tamil Nadu. He has published research papers in both national and International Journals and has authored 2 textbooks.