Journal of Computer Science

Original Research Paper

Framework for Enhancing the Performance of Data Intensive MPI based HPC applications on Cloud

Ashwini Janagal Padmanabha and Sanjay Harogolige Adimurthy

Department of ISE, Nitte Meenakshi Institute of Technology, Bangalore, Karnataka, India

Article history
Received: 29-04-2017
Revised: 13-06-2017
Accepted: 4-08-2017

Corresponding Author:
Ashwini Janagal Padmanabha
Nitte Meenakshi Institute of Technology, Bangalore, Karnataka, India
Email: ashwini.janagal@gmail.com

Abstract: Cloud computing is a new technology which is revolutionizing the current business model with pay-per-usage resource provisioning method. This model proves to be more profitable compared to traditional resource procurement and maintenance model. Data intensive High performance applications (HPC Application) handles large scale data sets on cluster/grid environment for enhanced performance. Most of these applications belong to the MPI category, where the work is assigned to multiple processes which communicate amongst each other to furnish the task. These applications prefer cluster/grid environment because of the homogeneity and high end resource availability. Cloud can be a better platform for these applications, as it consists of large quantity of resources. But, this technology is avoided by the HPC user community for the reasons of performance degradation, which is caused by the virtualization layer and sharing of resources. Static cluster instances as a resource provided by many cloud vendors like Amazon, CDAC etc. provides good performance by sacrificing the resource utilization factor. The work proposed here provides a framework for enabling data intensive MPI based HPC applications on cloud with dynamic cluster formation. Placement of the virtual machines hosting the individual processes and their distance to the data plays an important role in deciding the performance of application, as data transfer delay plays an important role in deciding the speed of execution. The framework provides two VM scheduling strategies towards improving the performance of data intensive HPC applications. The strategies with prioritized shared memory based communication of data to the process is implemented and tested on the private cloud. The work considers two most widely used data distribution models; Distributed volume and Striped volume. First VM scheduling strategy is implementable for distributed volume where complete data file will be hosted on single data server and the results show an improvement of around 88% in the best case. Second VM placement strategy can be used in more fine tuned distribution where stripes of single data file is distributed across different data servers. Here we have observed around 70% improvement in the performance of application compared to normal VM placement methods.

Keywords: Cloud, Data Intensive HPC, VM Placement, Data Distribution

Introduction

Large scale data intensive computing plays an increasingly important role in many scientific computations. Performance of these applications depends mainly on data locality, resource availability and data access pattern of the application. Parallel processing has been considered as the best solution for these applications, which are highly sensitive to performance. Many of the high performance based applications belong to the message passing (MPI) category and implemented through cluster/grid. Cluster/grid is set of tightly or loosely coupled machines which are almost homogeneous in nature.

Cluster/grid instantiation requires a large amount of resources, which makes it a costly solution. Cloud can be

considered as a solution for instantiating the clusters because of its resource abundance and scalable nature. Acquisition of the resources is quite easy and cheap on cloud platform. But, virtualization which is the base technology used in cloud implementation, degrades the performance of applications. For this reason HPC applications are avoided on cloud platform. But (Phillip and Andrzej, 2011) and (Guohui Wang, 2010) provides insight on few cluster-as-a-service cloud vendors for implementation of HPC applications on cloud. Static cluster instances are provided as a solution and the results indicate almost no degradation in the application performance but less resource utilization.

Ashwini *et al*. (2017a; 2017b) we have proposed a methodology to implement the dynamic clusters for communication intensive MPI based HPC applications with enhanced performance through resource reservation and VM scheduling policies. But for large scale data intensive applications, data availability will be the main criteria in deciding the performance. Work proposed here aims at enhancing the performance of data intensive MPI based HPC applications on cloud.

Current applications prefer distributed data model and the data location with respect to application depends on the distribution model. The individual processes in data intensive MPI applications, spend initial time in acquisition of the data which will be followed by processing. Porting these applications to cloud require a strategy to distribute the data such that data acquisition time will be reduced. The proposed framework enhances the performance of MPI based data intensive HPC applications by scheduling the virtual machines in accordance to the data availability.

Ashwini *et al*. (2017b), we have shown that shared memory based communication between VMs improves the VM communication performance drastically. Shared memory based I/O is implemented in this work, which gives better performance when compared to traditional TCP/IP based data communication. Data location with respect to the processing node is important in order to induce shared memory based I/O of data to process. Also in distributed data environment mapping of the data chunk to its respective process is very important. This work consists of a framework for data distribution on cloud platform and different VM placement strategies in accordance to the data location.

VM scheduling policies for two most widely used data distribution models; distributed volume and striped volume are proposed in this work. In distributed volume complete data file will be placed on single data server amongst the many servers. In striped distribution model, single data file will be divided into multiple chunks which will be distributed over many servers. The work is implemented and tested on private OpenNebula based cloud with GlusterFS distributed file system. The scheduling strategies strive to place VMs such that shared memory based I/O can be implemented. Results show an average enhancement of around 88% in case of distributed volume and 70% in case of striped volume by application of our VM scheduling strategy.

Rest of the article is organized as follows. Section 2 discusses some of the previous work which motivated us to consider this work. Proposed framework is discussed in section 3 and section 4 gives experiments conducted and result analysis. The work is concluded in section 5.

Previous Work

High performance applications are implemented through cluster/grid environment. Most of the existing work towards porting these applications on cloud platform consists of static cluster instances provided as a service by the cloud vendors through dedicated resources. Ekanayake and Fox (2009; Masud, 2010; Younge *et al*., 2011; Phillip and Andrzej, 2011; Nanos *et al*., 2007; Guohui, 2010) compare the performance of HPC applications on cloud/virtualization technology. Resource sharing amongst the virtual machines and the hypervisor layer are the main culprits in performance degradation of the applications on cloud. Some ready to use cluster-as-service solutions like Penguin on Demand (PoD), Amazon cluster instances etc. are discussed and compared with normal cluster performance in (Nanos *et al*., 2007) and (Guohui Wang, 2010). These cloud vendors provide static cluster-as-a-service instances consisting of homogeneous resources connected through high end networking. Results show a uninterrupted performance with low resource utilization. Any solution with less resource utilization will be costly for the cloud vendor as well as user.

Authors in (Jin *et al*., 2012; Shan *et al*., 2008; Mazandarani and Momeni, 2013; Rogeiro *et al.,* 2017) believe that capability of running HPC applications on top of data-intensive file systems is a critical catalyst in promoting Clouds for HPC. In (Shan *et al*., 2008) authors study many traditional data intensive applications and their performance on current super computing platform. The current supercomputing platform poses many challenges to these applications and they believe that separate benchmarks are required to evaluate these architectures. Jin *et al*. (2012) proposes a Chunk-Aware I/O (CHAIO) strategy to enable efficient N-1 data access on data-intensive distributed file systems. Authors in (Mazandarani and Momeni, 2013) propose QOS aware scientific application scheduling algorithm for cloud environment. Complete application is divided into work flow and according to QoS requirement such as cost or time resources are selected. This work does not consider the MPI based applications. Hou *et al*. (2016) authors study the I/O behavior on cloud. The experiments were conducted to measure various issues like effect of bandwidth, chunk size etc. on the I/O performance. The results show that various optimizations on I/O like enhancing the

bandwidth, communication latency etc. can be used to enhance the overall performance of the application.

There are few works towards study of performance of HPC applications on virtualization technology (Nanos *et al*., 2010; Graham and Shipman, 2008; Cheng and Wang, 2013; Ashwini *et al*., 2017a; 2017b; Hou *et al*., 2016) and (O'Donnacha *et al*., 2016). Porting of the HPC applications on cloud must consider performance degradation due to resource sharing, which leads to communication delays (Hou *et al*., 2016; O'Donnacha *et al*., 2016). For performance enhancement some of the best solutions proposed are, traffic shaping, shared memory based communication and VMM split driver models. Cheng and Wang (2013; Ashwini *et al*., 2017a) and (Ashwini *et al*., 2017b) shared memory based communication is used improve the performance. Results shows notable increase in performance with shared memory based communication.

Data distribution leads to more secure and loss resistant architecture. It is the most widely used architecture in the current scenario. Huang and Begnum (2013) proposes a layered, redundant data management approach for cloud-agnostic disaster recovery. This work shows the importance of data distribution and discusses the ways to recover them in cloud architecture. Noronha and Panda (2008) provides introduction to GlusterFS which is a distributed network file system. The work analyzes the performance of this system with consideration of various architectures. GlusterFS provides a client server architecture with servers called as bricks storing the data file in various formats. GlusterFS client can access the data from these servers. It is most widely used because of its simple architecture and ease of use. Therefore GlusterFS is used in the implementation of proposed system to form data distribution.

Manjaly and Jisha (2009; Sempolinski and Thain, 2010) and (Robinson and Hacker, 2012) authors have compared different cloud environments like OpenNebula, VMware Vsphere, Eucalyptus and Nimbus. Authors have analyzed the overall structure of each of these projects and addressed how the differing features and implementations reflect the different goals of each of these projects. Lastly, they discuss some of the common challenges that emerge in setting up any of these frameworks and suggest avenues of further research and development. Proposed work is implemented and tested in OpenNebula cloud management tool, which enables us to plug in our placement strategy.

Proposed Framework

The proposed work in Fig. 1 is aimed at enabling data intensive MPI based HPC applications on cloud. In general cloud data center consists of multiple data and compute nodes. GlusterFS distributed file system on private cloud platform is used in this work for establishing a distributed data base.

The work was initiated with the study of various data placement strategies and its effect on data transfer delay. The results proved that application performance can be improved, if data is near to the computing node in terms of transfer delay. Also, it has been observed that combining data and compute nodes will help in achieving good performance. The work considers a cloud platform, where individual hosts work as both data node and compute node. The work includes a data distribution module and VM placement module. In the first module, data will be hosted on the node according to the choice of user. In the next module, VMs hosting the data intensive MPI application are placed on the nodes such that the data transfer delay will be minimized.

There are various data distribution models. This work considers two very widely used models; distributed volume and striped volume. In the first model, complete file will be hosted on single data server amongst the set of data servers. In the later model, file will be striped and chunks will be stored amongst multiple data servers.

MPI based application hosting on cloud includes instantiating a cluster of VMs hosting individual processes. The work considers one process per VM for cluster formation. Two different VM scheduling policies have been proposed for the two different data distribution models. The scheduling policies implemented, enhances the application performance through prioritization of the shared memory based I/O by placing VMs according to the data dependency.

Data Availability Vs. Application Performance

Data placement is the key candidate in deciding the performance of data intensive HPC applications. In order to understand the effect of data locality on performance of application the work was initiated by the study of various scenarios of data distribution and its effect on the performance of application. Data repository can be divided into two categories, as centralized or distributed. Centralized data bases keep data on single node or single rack whereas distributed data bases tend to distribute the data over different locations. In the proposed method, we have implemented and tested an environment wherein data is distributed amongst the processing nodes. As a first phase, we tested the impact of data distribution on the performance of application using IOR benchmark application. Following cases were considered for our analysis:

- Application with single process and Data file hosted on same node
- Application with single process is hosted on a compute node and file should be transferred from separate data node
- Multi process application hosted on cluster of compute nodes and data file transferred from single data node
- Multi process application hosted on the nodes consisting of data file

DOI: 10.3844/jcssp.2017.320.328

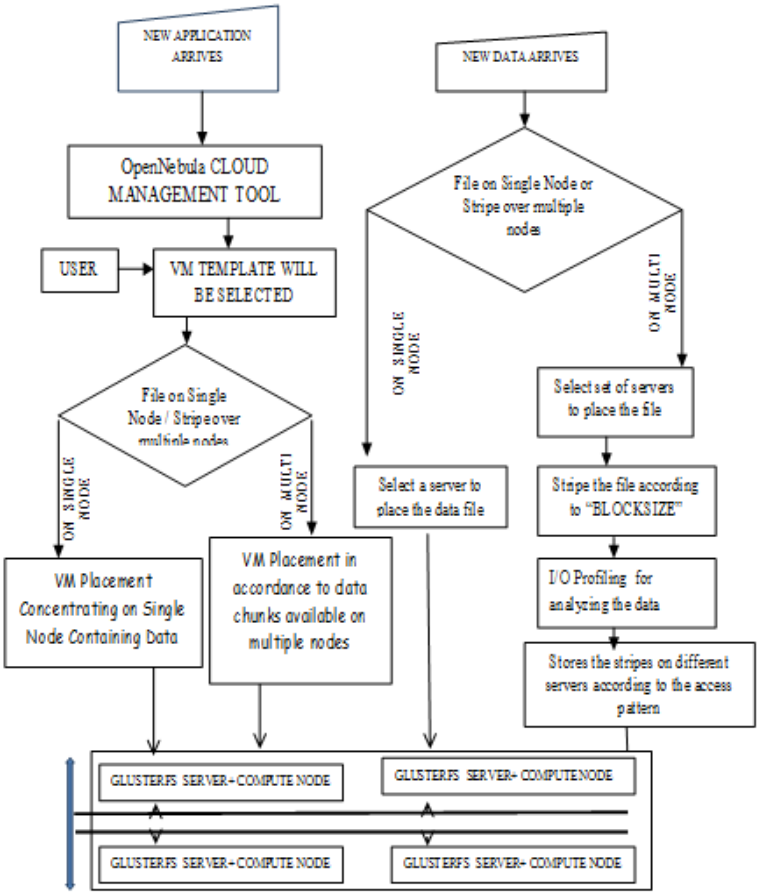

Fig. 1. Proposed framework

The study of various cases helped us to understand the importance of data transfer time in overall performance of application.

Application with single process and Data file hosted on same node.

MPI applications will be executed with multiple processes and the number of processes is usually specified by the user. In this scenario application was executed using single process on the virtual machine hosted on the node containing data file. This scenario doesn't explore the parallelization, but it analyzes the advantage of holding process and data on same machine, which reduces the data transfer latency due to shared memory based communication.

Application with single process is hosted on a compute node and file should be transferred from separate data node.

Most of the cloud frameworks consider data and compute nodes separately with data transfer during the beginning of execution. Separate data nodes help in better management of data. In this scenario, the application was run as a single process accessing file from data node. Even though both data node and compute node are mounted on same rack, formal TCP/IP based communication incurs latency.

Multi process application hosted on cluster of compute nodes and data file transferred from single data node.

As per the general cloud practice, file was hosted on data node and virtual machines hosting the individual process were placed on separate compute nodes. All the processes download data file and work on their respective chunk required. This scenario increases the data acquisition delay.

Multi process application hosted on the nodes consisting of data file.

Every workstation was designed as GlusterFS file server and data files were striped according to the “Block size” and chunks were stored on servers in increasing order of IP address. Application along with number of processes equivalent to number of chunks of the requested data file were hosted on the machines according to their data requirement. For example process 0 hosting VM was hosted on GlusterFS server containing first chunk and so on. Here, data acquisition time will be reduced as data and process on same host communicate through shared memory based I/O.

DOI: 10.3844/jcssp.2017.320.328

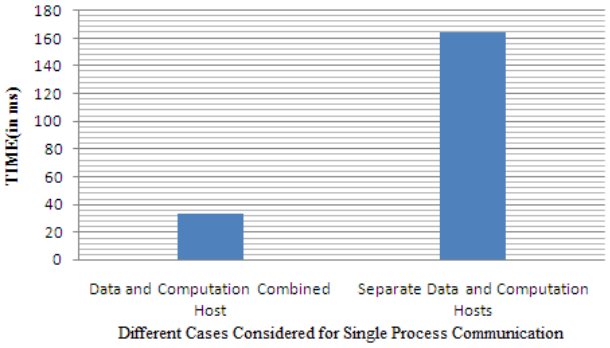

Fig. 2. Graph to compare the time of execution of IOR application with single process on different scenarios

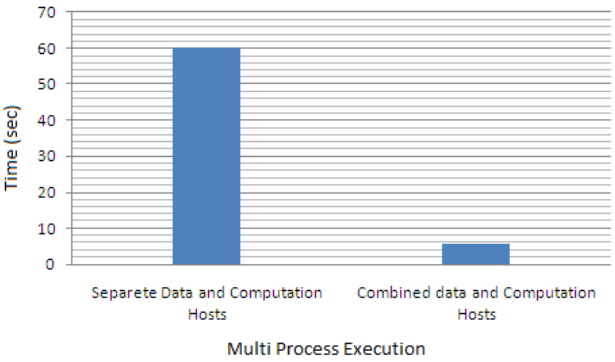

Fig. 3. Graph to compare the time of execution of IOR application with multi process on different scenarios

The four cases were analyzed using a sample data file of size 13GB. Figure 2 shows that for single process application, combined data and compute node gives better performance compared to separate nodes. We can observe around 82% of performance enhancement in combined data and compute node scenario compared to separate nodes. Figure 3 shows that for multi process application also combined data and compute nodes gives better performance.

For multi process scenario we have analyzed the results using number of processes as 13. For separate data and compute node, data is stored on single server and transferred to the VMs. File is divided into block size of 1GB for the combined data and compute node and distributed over volumes in three workstations. Here number of virtual machines is equivalent to number of processes with one process per VM. We can observe around 92% of performance enhancement in case of combined data and compute node (Fig. 3). Major part of this performance improvement goes to the reduction in data transfer delays. The results made us to conclude that the combined data and compute nodes gives better performance. But, the major problem we faced here is the data dependency and process to data mapping.

Virtual Machine Placement for Data Intensive HPC Applications

Proposed model aims at creation of dynamic VM clusters on cloud, for data intensive MPI based high performance application. In this regard, objective of this work is to place the virtual machines in accordance with their data dependency, in order to reduce their data transfer delay. The analysis done in previous section showed us that maximum performance can be achieved by application, when data is available on the same host for the application. This made us conclude that a cloud framework, wherein, data and processing node functionalities are combined will enhance the performance of data intensive applications on cloud. The proposed work set up a distributed data model on compute nodes for the data used by the HPC applications. For other applications data nodes will be hosted separately. In order to implement this, GlusterFS distributed file system was implemented in all the processing nodes. Placement algorithm proposed searches for the host where data file is available and try to fit the virtual machine hosting application on the same machine. There may be situation where in host machine with data is not having enough resources to host the virtual machines. Then placement strategy will try to fit the VM in a host machine in the same rack where data is hosted. The dynamic cluster of virtual machines hosted through this will ensure complete resource utilization with good performance, unlike static cluster instances provided by the cloud vendors with dedicated compute and network resources which leads to the wastage of the resources.

As mentioned in the previous section, we have considered distributed volume and striped volume amongst the various data distribution models. In the distributed volume, complete data file will be stored in same data node but overall database consists of many servers and file can be stored in any server depending on the space available. If the complete file resides on single node, then virtual machines that execute the processes should be placed in the node containing file. The scheduling policy proposed will search for the host with the requested file and try to place the maximum possible VMs on the host. In case of resources on the host not adequate for the instantiation of the VMs, any host in the same rack with maximum resources available will be selected for the remaining VMs. Algorithm 1 gives the proposed method.

Storing a file on single node will increase the load on node if many people are accessing the file at the same time. Also in case of node failure complete data will be lost. A better solution is to stripe the file into smaller chunks and store it in different nodes. In the striped volume model of data distribution, data file will be striped and chunks will be stored over different data

servers. This architecture has the advantage that if one server goes down, then only a part of data file will be missing. Also, some optimization can be applied for this architecture like an intelligent distribution technique where in chunks will be distributed to the nodes on same rack, so that their transfer time will be less.

In order to analyze the effect of striping on performance of application hosted on virtual machine, the file has to be chopped (striped) to particular size and distributed across the servers. The primitive placement strategy applied finds the set of servers, on which file strips are available and virtual machines hosting the application are hosted on those servers. For this scheduling to work properly, the algorithm should know the data dependency of each process. We have many MPI profiling tools like MpiP, Darshan etc, to study the data access pattern of each process. In this work we have considered the IOR application which divides data into different segments and each segment is divided evenly among the processes. For example, if number of processes is equal to number of segments then each process will handle one data segment respectively. If an unknown application is submitted by the user then the I/O profiling phase will decide the data distribution pattern Algorithm 2 will provide the steps for scheduling of VMs in case of striped data. We have assumed that the number of processes will be equal to number of blocks striped across.

*Algorithm 1. VM Scheduling In case of data file placed on single serve*r.
Data: Set of workstations 'S', Number of MPI Processes 'N' and Data File 'F'
Result: Optimal VM Placement
Initialization;
repeat
  Check server S*i* for F;
Until 'F' is found;
If 'F' is found then
  repeat
  repeat
    Schedule Virtual Machine V*i* in S*i*;
    Until (Resources are not available in S*i*) OR
        ('N' VMs are hosted);
    chose the next host machine in same rack and
     with maximum resources.;
  until ('N' virtual machines are hosted);
else Convey that data file is missing;
end

Experimental Setup and Results

The experiments are conducted on real private cloud setup. A private cloud with OpenNebula cloud management software was setup with three workstation working as both GlusterFS data node and compute node for hosting VMs. Xen hypervisor is used to create the virtual machines. OpenNebula is a completely open source toolkit to build any type of Infrastructure as a Service (IaaS) cloud. It manages various facets of cloud like storage, network, virtualization, monitoring and security. OpenNebula includes single front end, containing the management interface and a bunch of workstations, used for hosting data and VMs. The workstations are quad core with 24GB RAM and 250GB hard disks.

Algorithm 2. VM Scheduling in case of striped data files
Data: Set of workstations 'S', Number of MPI Processes 'N' and Data File 'F'
Result: Optimal VM Placement
Initialization;
repeat
  Check Set of Servers CS, where chunks of F found
Until 'F' is found;
If 'F' is found then
  repeat
    if Resource is available at CSi and matches the
      Data Dependency then
      Schedule Virtual Machine Vi in CSi
      else
      Schedule Vi in the host available in same
      rack as of data and with maximum available
      resources;
    end
  until ('N' virtual machines are hosted);
else
 Convey that data file is missing;
 end

In order to implement and test the proposed scheduling method GlusterFS (Graham and Shipman, 2008) scale-out network-attached storage file system is used. GlusterFS is a scalable open source file system, capable of scaling to several peta bytes and handling thousands of clients. GlusterFS is widely used for its implementation simplicity, with minimal cost. GlusterFS, clusters different disks and memory resources and manage the data in a single global namespace.

GlusterFS defines multiple volumes and distributes files across them, essentially making one single larger storage volume from a series of smaller ones. Distributed Gluster volumes are used to scatter files randomly across the number of bricks in the volume. Type of a volume is specified at the time of volume creation and volume type determines how and where data is placed. Following are the two widely used volume types supported in GlusterFS:

- Distribute: Complete data file will be places on single data server amongst many data servers available

- Stripe: Striped volume divides data into stripes called chunks and each chunk will be stored on different server

Figure 4 show the Gluster file system implementation in the proposed system.

IOR HPC benchmark application is used with a simple data file of size 13GB. This benchmark application is used to test parallel file systems using various interfaces and access patterns. The benchmark application works in two modes. In the first one, all processes work on entire data file. In the later mode, the data file will be divided into different blocks. Blocks will be allotted evenly to the processes. The process with rank 0 gets the first block and the process with rank 1 gets the second blockand so on. Each block is further divided into many transfer units called Transfer Size, which is the data transferred for a process between memory and file for each I/O function call. The memory buffer size is equal to the Transfer Size.

Distributed database is implemented using GlusterFS file system and all the workstations work as both data node and compute nodes. For the first scheduling policy distributed file system of GlusterFS is used and data file as a whole will be placed on single server and VMs will be scheduled on same node. For the second scheduling, data file will be striped into chunks and it will be distributed across the servers. OpenNebula cloud management tool uses matchmaking scheduling policy as default for VM placement and this policy has high priority for resources and neglected the remaining aspects of an application.

For algorithm 1, IOR-HPC application was run with number of VMs equivalent to the number of processes and VMs were hosted on the machine containing the file. This scheduling policy is easy to implement but suitable for the case, where data files are not striped to fit in different servers. If the file is only single copy of the file this policy will try to fit all the VMs in the same host containing file. In case of inadequate amount of resources, VMs will be fitted in the host with maximum resource availability on the same rack. Replication of the file can give more choices for the VM placement and in case of replicas we place VMs in decreasing order of resource availability. We can observe that the application performs around 88% (Fig. 5) better in case of all the VMs hosted on data node compared to matchmaking policy, where VMs are placed only on the basis of more resource. Even in cases where data nodes were not having enough spaces and few VMs are hosted on same rack as of data node we can observe an improvement of around 66% (Fig. 5).

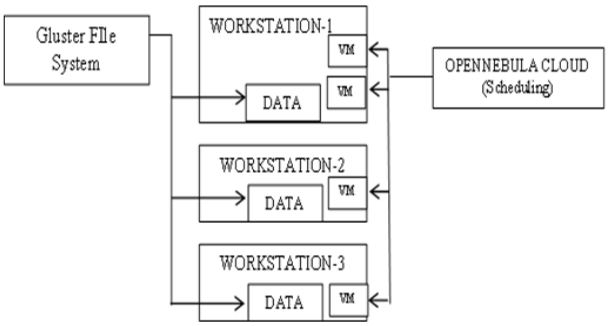

Fig. 4. Private cloud with merged data and compute

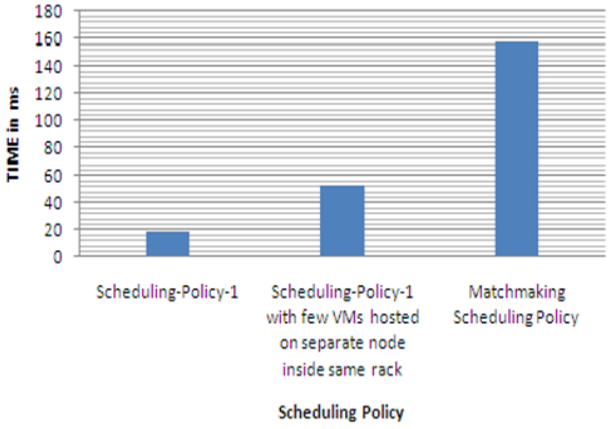

Fig. 5. Graph to compare the time of execution of IOR application executed with scheduling algorithm 1 for VM placement and default resource based scheduling policy

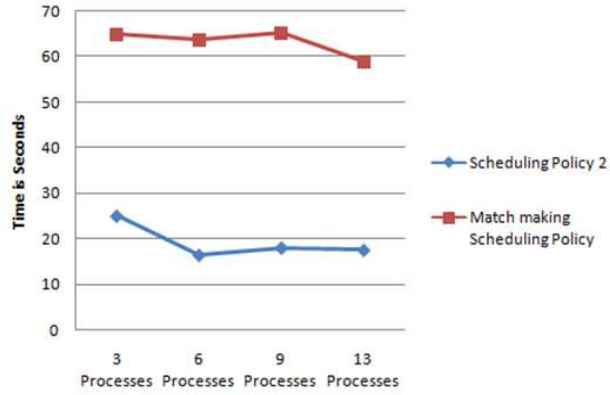

Fig. 6. Graph to compare the time of execution of IOR application executed with scheduling algorithm 2 for VM placement and default resource based scheduling policy

This algorithm gives good performance and easy implement but the level of distribution is very coarse. Entire data file on single system will be a great risk in case of system crash. Algorithm 2 will be used in case of fine grained distribution. Here, data file is striped and

stored in different servers and VMs should be placed in the server with data chunk, according to their data dependency. IOR-HPC application works in two modes. In the first mode, entire file will be processed by each process and in the second one, each process requires a particular block size and processes are sequentially mapped to block. For example, block 0 is mapped to process 0, block 1 to process 1, etc. We have striped data according to the block size and VMs hosting the individual process are scheduled in the data node containing required block. Analysis has been done for various numbers of processes. Consider the result analysis for the sample processes 3, 6, 9 and 13. For separate data and compute node, data is stored on single server and transferred to the VMs. File is divided into block size of 1GB for the combined data and compute node and distributed over volumes in three workstations. Here number of virtual machines is equivalent to number of processes with one process per VM. Also, in all the four cases we are able to fit in the VMs according to their data dependency except when number of processes is 3. In this case as very less VMs are there, few chunks have to be transferred across the workstations. Graph in Fig. 6 shows that on an average there is around 70% of improvement in performance of our placement strategy, to default resource based strategy.

Conclusion

HPC applications usually work with a large amount of data and high performance resources. Even though cloud can provide plenty of such resources, it is not much entertained for HPC applications. Reason is performance degradation due to sharing of resources and hypervisor layer. Current database technology is driven by distributed databases. The work has implemented a framework for improving the performance of data intensive HPC applications on cloud, by redesigning the VM placement policies. Our analysis shows that much of the performance degradation is while transferring data to processes. Usually cloud technology, differentiates data, compute nodes and as an initial step data will be transferred from data node to compute node. Our experiments shows that combined data and compute nodes give better performance, as data transfer delay will be minimized. Two VM placement strategies for distributed and striped distributed data models are implemented and tested in this work, using GlusterFS distributed file system and OpenNebula based private cloud. The placement policies strive to instantiate the VMs running the process on the host machine where data is already residing. For distributed data model our strategy has achieved around 88% of performance enhancement and 70% enhancement for striped distributed model.

Acknowledgement

The authors are indeed grateful to the management of Nitte Meenakshi Institute of Technology for their support and resources provided to us at every stage of this work.

Funding Information

We want to express our immense gratitude to the Department of Electronics and Information Technology, New Delhi for funding this work (Administrative Approval number: HPC Projects/2(8)/2011).

Author's Contributions

Ashwini Janagal Padmanabha: Being the research scholar, author has contributed in design of the work, data collection, implementation, result analysis and drafting of the article.

Sanjay Harogolige Adimurthy: Being the research advisor, author has contributed in the conception of the work, result analysis and critical revision of the work and article.

Ethics

As per the knowledge of the authors, work carried out in this article is unique.

References

Ashwini, J.P., H.A. Sanjay and G. Shreevidya, 2017b. Modeling and prediction of bandwidth requirement for MPI based HPC applications on cloud. Int. J. Cloud Comp.

Ashwini, J.P., H.A. Sanjay and M.C. Naina, 2017a. Framework for performance enhancement of MPI based HPC Application on Cloud. Int. J. Grid High Performance Computing.

Cheng, L. and C.L. Wang, 2013. Network performance isolation for latency-sensitive cloud application. J. Future Generation Computer Systems, 29: 1073-1084. DOI: 10.1016/j.future.2012.05.025

Ekanayake, J. and G. Fox, 2009. High performance parallel computing with clouds and cloud technologies. Proceedings of the 1st International Conference on Cloud Computing, Oct. 19-21, Munich, Germany, pp: 20-38. DOI: 10.1007/978-3-642-12636-9_2

Graham, R.L. and G. Shipman, 2008. MPI support for multi-core architectures: Optimized shared memory collectives, recent advances in parallel virtual machine and message passing interface. Proceedings of the 15th European PVM/MPI Users' Group Meeting, Dublin, Ireland.

Guohui Wang, T.S., 2010. The impact of virtualization on network performance of Amazon EC2 data center, INFOCOM, San Diego, CA.

Hou, B., F. Chen, Z. Ou, R. Wang and M. Mesnier, 2016. Understanding I/O performance behaviors of cloud storage from a client's perspective. Proceedings of the 32th Symposium on Mass Storage Systems and Technologies, May 2-6, IEEE Xplore Press, USA, pp: 1-12. DOI: 10.1109/MSST.2016.7897089

Huang, K. and K. Begnum, 2013. The Hydra: A layered, redundant configuration management approach for cloud-agnostic disaster recovery. Proceedings of the IEEE 5th International Conference on Cloud Computing Technology and Science, Dec. 2-5, IEEE Xplore press, UK. DOI: 10.1109/CloudCom.2013.158

Jin, H., J. Jiayu, X.H. Sun, Y. Chen and R. Thakur, 2012. CHAIO: Enabling HPC Applications on Data-Intensive File Systems. Proceeding of the 41st International Conference on Parallel Processing, Sept. 10-13, IEEE Xplore press, USA. DOI: 10.1109/ICPP.2012.1

Manjaly, J.S. and S. Jisha, 2009. A comparative study on open source cloud computing frameworks. Int. J. Eng. Comp. Sci., 2: 2026-2029.

Masud, R., 2010 High Performance Computing with Cloud. withCloudsRaihanMasud.pdf

Mazandarani, A. and H. Momeni, 2013. QOS-Aware scientific application scheduling algorithm in cloud environment. J. Comp. Eng. Intelligent Systems, 4: 21-30.

Nanos, A., G. Goumas and N. Koziris, 2007. Exploring I/O virtualization data paths for MPI applications in a Cluster of VMs: A networking perspective. Proceedings of the 2010 Conference on Parallel Processing Euro-Par, Aug. 31-Sep. 03, ACM, Italy, pp: 665-671.

Nanos, A., G. Goumas and N. Koziris, 2010. Exploring I/O Virtualization data paths for MPI applications in a Cluster of VMs: A networking perspective. Proceedings of the Conference on Parallel processing (Euro-Par'10), Ischia, Italy.

Noronha, R. and D.K. Panda, 2008. IMCa: A High Performance Caching Front-End for GlusterFS on InfiniBand, Proceedings of the 37th International Conference on Parallel Processing, Sept. 9-12, IEEE Xplore press, USA. DOI: 10.1109/ICPP.2008.84

O'Donnacha, F., E. Ragnoli, S. Venugopal, S.C. James and K. Katrinis, 2016. On the efficiency of executing hydro-environmental models on cloud. 154: 199-206. DOI: 10.1016/j.proeng.2016.07.447

Phillip, C.C. and G. Andrzej, 2011. IaaS Clouds Vs. Clusters for HPC: A Performance Study. Proceedings of the 2nd International Conference on Cloud Computing, GRIDS and Virtualization Cloud Computing, Sep. 25-30. Italy, pp: 39-45.

Robinson, N. and T. Hacker, 2012. Comparison of VM deployment Methods for HPC education. Proceedings of the 1st Annual Conference on Research in Information Technology (RIIT12), Calgary, AB, Canada.

Rogeiro, J., M. Rodrigues, A. Azevedo, A. Oliveira and J.P. Martins *et al*., 2017. Running high resolution coastal models in forecast systems: Moving from workstations and HPC cluster to cloud resources. Adv. Eng. Software Workstat. DOI: 10.1016/j.advengsoft.2017.04.002

Sempolinski, P. and D. Thain, 2010. Cloud computing technology and science. Proceedings of the IEEE 2nd International Conference on Cloud Computing Technology and Science (CloudCom' 10), IN, USA. pp: 1-4. DOI: 10.4018/ijghpc.2013100101

Shan, H., K. Antypas and J. Shalf, 2008. Characterizing and predicting the I/O performance of HPC applications using a parameterized synthetic benchmark. Proceedings of the ACM/IEEE Conference on Supercomputing, Nov. 15-21, ACM, USA.

Younge, A.J., R. Henschel, J.T. Brown, G. von Laszewski and J. Qiu *et al*., 2011. Analysis of virtualization technologies for high performance computing environments. Proceedings of the IEEE International Conference on Cloud Computing, Jul. 4-9, IEEE Xplore press, USA. DOI: 10.1109/CLOUD.2011.29