Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2023 SCPE. Volume 24, Issues 1, pp. 55–67, DOI 10.12694/scpe.v24i1.2086

FRAMEWORK FOR PERFORMANCE ENHANCEMENT OF MPI BASED APPLICATION ON CLOUD

ASHWINI J P*, SANJAY H A†, NAYANA M C‡, K ADITYA SHASTRY §, AND MOHAN MURTHY M K¶

Abstract. Cloud technology is a major revolution that has happened in the computing era that has changed the way applications and resources are used. Elasticity is the key characteristic of the cloud, wherein the required number of resources are provided as a service with a pay-as-you-go principle. This reduces the huge cost involved in buying, installing, and maintaining the resources. Cloud computing, with its highly scalable resources, can be a good platform for High-Performance Computing (HPC). HPC application performance highly depends on the quantity of the resources, which makes the cloud a suitable candidate. But the HPC community is not very happy with cloud technology, and most of the users still think cloud technology is not suitable for HPC applications. But virtualization technology, which is the foundation of the cloud, degrades the performance of applications in the urge to improve utilization. The hypervisor layer and resource sharing by the virtual machines (VMs) hosted on the same node are the main reasons for performance degradation in the cloud. The majority of the HPC applications belong to the message passing (MPI) category, and for these applications, communication cost is the major stakeholder in deciding performance. If these applications are hosted on the cloud, it leads to further performance degradation as the process on a VM communicates through the virtual network interface, which in turn shares the network interface of the host machine with other VMs. MPI-based applications hosted on MPPs work in a bandwidth shared environment as multiple processes communicate over the same network. But in the cloud, as the number of VMs increases, per node bandwidth availability per communication reduces. To address the above issues, we have built a framework to enhance the performance of MPI-based HPC applications on VMs by considering proper VM placement strategy and resource reservation policies, with knowledge of resource availability and process communication patterns. A VM placement strategy for dynamic clustering of VMs with high priority for shared memory-based communication is proposed and tested. Results show that with a medium number of processes, there is an improvement of around 70% with our placement strategy for high data communicating processes. If there are fewer processes, and the single physical node can hold all the VMs, then the performance improvement is up to 500%.

Key words: Cloud Computing, HPC, Heterogeneous resources, k-means Template Clustering, Ranking of Resources

AMS subject classifications. 68M14

1. Introduction.

Cloud computing has given rise to a new business model wherein hardware and software resources are procured as a service from cloud vendors and paid as per usage. This model reduces the burden of a large initial investment and resource management and maintenance costs. The main feature of the cloud is elasticity, wherein the resources can be scaled according to requirements. This makes the cloud an ideal solution for many kinds of applications. Virtualization is the core technology involved in cloud computing, which enhances resource utilization. But the downside of virtualization is that it sacrifices the application's performance. The virtualization layer and resource sharing are important culprits in reducing the performance of an application.

High-Performance Computing applications are used by a wide range of users like researchers, industry, etc. These applications are executed on a cluster/grid to gain maximum performance through parallelization. The main criteria for HPC applications is the quantity of resources available. The cloud can be a suitable platform for HPC applications considering the number of resources available at lower cost and ease of access. But as the performance of a VM is less compared to an actual physical machine, we avoid the cloud for HPC applications. Some of the solutions already existing in the market provide cluster-as-a-service wherein users will pay for

*Department of AIML, JNN College of Engineering, Shivamogga,India, 577 204 (ashwinijp@jnnce.ac.in).
†Department of ISE, MS Ramaiah Institute of Technology,Bengaluru, India, 560054 (sanjay.ha@msrit.edu)
‡Department of PG, Nitte Meenakshi Institute of Technology,Bengaluru, India, 560064 (nayanamc91@gmail.com)
§Department of ISE, Nitte Meenakshi Institute of Technology,Bengaluru, India, 560064 (adityashastry.k@nmit.ac.in)
¶IBM, USA, (maakem@gmail.com)

static cluster instances. They believe that HPC applications perform better only with dedicated resources [1–2]. These solutions are costly from the perspective of cloud vendors, as the study [3] shows that the utilisation of resources is very low in these static cluster instances.

A good solution for these problems could be dynamically creating a cluster of VMs based on user requirements and promising adequate resources based on application requirements. Most HPC applications belong to the message passing (MPI) category, where multiple processes communicate with each other to solve a problem. Communication costs are the main bottleneck in deciding the performance of these applications. According to [4], the communication performance of VMs is very low because every communication goes through a virtual network interface to the network interface of the host machine, which is shared by all VMs on that machine. Along with this, the hypervisor layer adds some extra delay for VMs to access hardware. As the number of VMs hosted on a node increase, resource availability decreases. It has been observed that [4] VMs communicating through shared memory achieve high performance compared to classic TCP/IP based communication. In this work, we propose a framework for enhancing the performance of VMs hosting MPI-based HPC applications by combining communication aware scheduling with the reservation of adequate network bandwidth. The proposed framework provides a VM placement strategy to set up a dynamic cluster of VMs hosting individual processes of HPC applications with shared memory-based communication or bandwidth reservation. Results show that with intelligent placement and resource reservation, performance is improved. If the number of processes is less, then with pure shared memory-based communication, performance is enhanced by up to 500% when compared to a normal scheduling policy. With a medium number of processes wherein some of the processes are communicating with traditional TCP/IP based communication, performance is improved by up to 70%.

The rest of the paper is organized as follows. In section 2, we discuss important works done in this area. Section 3 explains the proposed framework. Section 4 gives details about the implementation and results, followed by the conclusion.

2. Related work. In [1], the authors Jisha S et al. focus on different frameworks for cloud computing. Cloud computing is a developing area that allows users to deploy applications with better scalability, availability, and fault tolerance. Cloud computing focuses on delivering virtual network services so that users can access services anywhere in the world with the necessary quality of service requirements. Cloud computing is a technique where resources are accessed, and services are needed to perform functions on-demand.

The work done by Peter Sempolinski et al. in the paper [2] is focused on the comparison between different cloud environments like OpenNebula, Platform ISF, VMware Vsphere, Eucalyptus, and Nimbus. Begin with a short summary comparing the current raw feature sets of these projects. After that, deepen the analysis by describing how these cloud management frameworks relate to the many other software components required to create a functioning cloud computing system. They also analyze the overall structure of each of these projects and address 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. These include the problems of fair scheduling in the absence of money, eviction or preemption, the difficulties of network configuration, and the frequent lack of clean abstraction. The work done by Nicholas Robison et al. in [3] is focused on describing experiences with using virtualization for virtual high performance computing clusters for education and compares the performance of the popular OpenNebula virtualization manager using both NFS and SSH for virtual machine image sharing. Their results show it is possible to develop an effective teaching environment using commodity desktop computers and network hardware along with open-source virtualization software. The work done by Nan Li et al. [4] is focused on a comprehensive survey, with sufficient analysis, of current inter-domain communication mechanisms on Xen-based hosting platforms. Techniques proposed by recent researchers to enhance communication performance are compared and discussed from three perspectives, which are locations, access, and management of shared memory regions. In addition, detailed overviews of various topics regarding virtualization.

In [5], Dhabaleswar K. Panda et al. focus on the following three steps to demonstrate the ability to achieve near-native performance in a VM-based environment for HPC. First, they have proposed Inter-VM Communication (IVC), a VM-aware communication library. It will support efficient shared memory communication among computing processes that are on the same physical host, even though they may be in different VMs.

This is critical for multi-core systems, especially when individual computing processes are hosted on different VMs to achieve fine-grained control. Second, they have designed MVAPICH2-ivc, a VM-aware MPI library based on MVAPICH2 (a popular MPI library), which allows HPC MPI applications to transparently benefit from IVC. Finally, they evaluate MVAPICH2-ivc on clusters featuring multi-core systems and high-performance InfiniBand interconnects.The work done in [6] is focused on integrating HPC interconnect semantics into the VMM split driver model. They aim to decouple data transfers from the virtualization layers and explore direct application-to-NIC data paths. Nonetheless, the implications of this mechanism on the overall throughput constitute a possible caveat of their approach: the way the control path interferes with data communication may result in significant overhead. To justify developing a framework to support standard HPC interconnect features (user-level networking, zero-copy, and so on) in VM environments, we must first investigate the behaviour of HPC applications in such environments. Hence, they deploy network benchmarks and a real scientific application in a cluster of Para virtualized Xen VMs and present some preliminary results.

With the work done by Richard L. Graham and Galen Shipman in [7], with local core counts on the rise, taking advantage of shared memory to optimize collective operations can improve performance. The authors studied several on-host shared memory optimised algorithms for MPI-Bcast, MPI-Reduce, and MPI-Allreduce, using tree-based, and reduce-scatter algorithms. In [8], the authors propose an improved K-means data clustering algorithm. We use the method based on this work to create a cluster of computing resources. This paper focuses on building a strategy to cluster the resources (VMs) for efficiently running MPI applications by building a degree of similarity function based on user requirements.

Most of the solutions provided by cloud vendors for HPC applications include provisioning of static clusters. They end up paying a high price in terms of resource wastage. Our work aims at the deployment of dynamic clusters for HPC applications without compromising performance. The main objectives of this work are:

- Grouping the VM templates based on CPU and memory values to provide a homogeneous environment.
- Ranking the host machines according to the amount of free CPU and memory available.
- Placement strategy for VMs to enhance the communication performance.
- Network resource reservation for VMs on different hosts to communicate.

3. Proposed Work. The proposed framework is shown in Fig 3.1. It aims to enhance the performance of HPC applications on cloud platforms by communication aware VM placement and reservation of resources.

Virtual machines are usually hosted in accordance with the topology of the data center or scheduling policies like round robin, matchmaking, etc. A user requests a virtual machine with his requirements for CPU cores and memory. If a virtual machine template with the same characteristics as the user requirement is available, then it will be chosen, or a new template will be selected. Amazon divides the VMs into different categories, like small, medium, large, and extra-large. This solution is easy to implement, but there is a high possibility of mismatch between the selected template and user requirements, which may lead to either resource wastage or application performance degradation.

Therefore, in the proposed framework, we provide the template exactly as per the user's requirement. The first step is to search for the template. Searching for the required one in the complete template database is quite time consuming. For this purpose, the first module of the framework is to group the VM templates according to CPU and memory values. Grouping will localise the search to a group rather than a complete database. Every time a new template is added, grouping modules need to be executed. After selecting the VM template, the next step is to host it. As the work considers only MPI-based HPC applications, we have considered communication resources as being of the highest priority, along with processing and memory resources. In this regard, the placement strategy gives the highest priority to shared memory-based communication when hosting the virtual machines that are part of the cluster. For this purpose, we need to rank the machines that host the virtual machines depending on various resource availability factors. The CPU and memory requirements are provided by the user, and consistent provisioning of the requested quantity needs to be maintained according to the SLA. A strategy to rank the machines according to the availability of multiple resources is used, and VMs are hosted in descending order of their rank.

As the rank of the machine changes with applications hosted on it, this module must be executed every time a new request comes from a user. It is impractical to think that we will be able to host all virtual machines on a single host so that they can take advantage of shared memory communication. The bottleneck in the overall

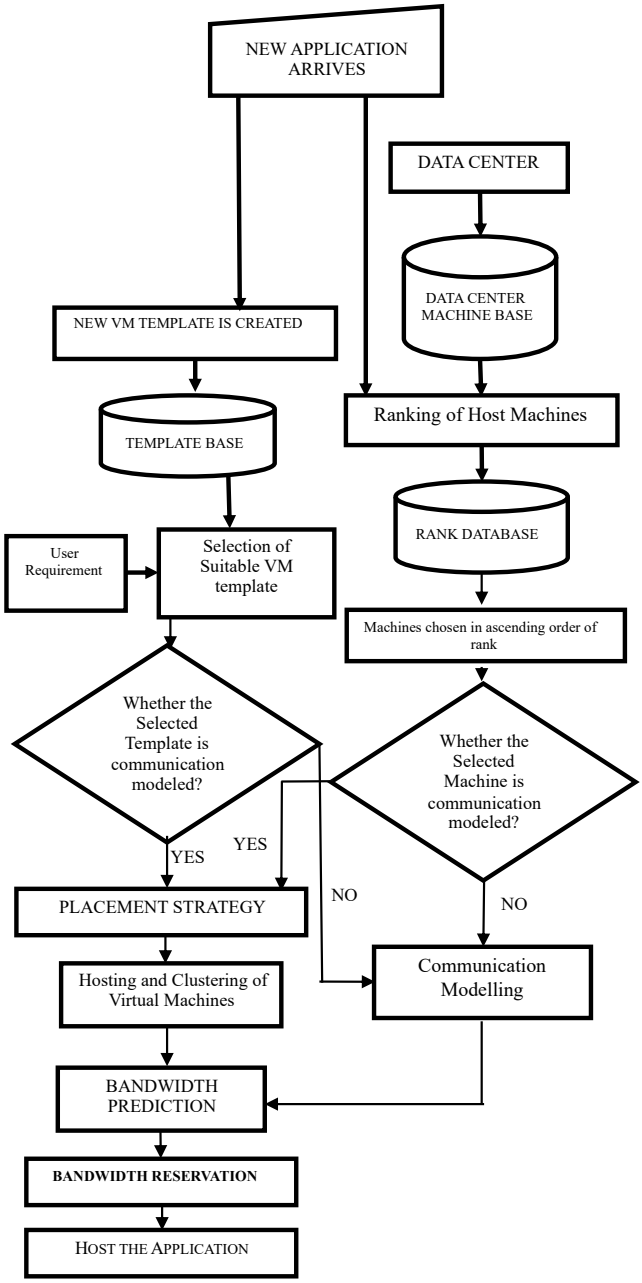

Fig. 3.1: Proposed Framework

performance of the application will be the VMs that are part of the cluster but hosted on different nodes. In [3], we have proposed a bandwidth modelling and prediction framework for MPI based HPC applications on the cloud. This framework is specific to the size of data exchanged amongst processes and the hosting of VMs and node machines.

So, every time a new VM template is created, it must be modelled for bandwidth prediction. Also, whenever a new host machine is added to the data center which is a very rare event to happen, it must be modelled. It has been observed that the reservation of the network bandwidth for the applications drastically improves

the performance. Predicted bandwidth will be reserved between clustered VMs that are hosted on different machines. Reservations make sure that communication between VMs will not fluctuate in accordance with the change in available bandwidth.

3.1. Grouping the VM templates based on CPU and Memory values. Starting a VM from scratch takes a long time. For this reason, cloud vendors provide preconfigured virtual machine images called virtual machine templates. In a normal cloud, users choose VM templates and the required number of VMs for that template will be instantiated.

In [9], we worked on clustering the virtual machines on the cloud. The K-Means algorithm is used to cluster the virtual machines dynamically based on a similarity metric (execution time of a benchmark application). According to the results, the best clusters are made up of virtual machines with similar capabilities and are also more powerful than those with different characteristics. Once the best clusters are obtained, the most efficient machines based on the user's requirements in the best clusters can be chosen considering network parameters. However, when we attempted to implement the work in a real-world cloud environment, we realized that it was impractical because VMs would not be available in a running state. They must choose the VM templates that will be instantiated.

Cloud vendors like Amazon divide the templates into three to four different categories, and users must select one from amongst these. But provisioning of the VM as per user requirements will improve the utilization and reduce the cost. The template base is a large collection of all the VM templates available in the cloud and searching for the user-requested template in a huge database takes time. To save time in searching for the requested template, we group the VM templates according to the CPU and memory values as a similarity metric. Grouping will localize the search to a particular group rather than the complete database. Each application varies in its priority towards usage of processing elements and memory. Some require a lot of CPU power, while others require a lot of memory. Depending on the priority towards CPU and memory, each template is marked with a weightage using the equation 3.1.

$$\begin{aligned} TemplateWeightage = CPU_{PRIRT} \times \\ \frac{CPU - value - given - for - VM - Template}{Total - Number - Of - Physical - CPUs - In - Single - Workstation} \\ + MEM_{PRIRT} \times \frac{Memory - Value - Given - For - VM - Template}{Total - Memory - Of - Single - Workstation} \end{aligned} \tag{3.1}$$

K-means algorithms are applied to group the templates according to their weightage. When a new requirement arises, we locate the group with similar CPU and memory values as the required one. And our search will remain local to that group. The K-means algorithm is a prototype based partitioning technique that attempts to find a user-specified number of clusters (K), which are represented by their centroids. The K-Means algorithm takes the input parameter K provided by the administrator and partitions a set of 'N' virtual machine templates into 'K' clusters based on "template weightage" so that the resulting intra-cluster similarity is high, but the inter-cluster similarity is low.

Algorithm 1 VM Grouping based on CPU using KMeans

1: Calculate template weightage for all the N templates
2: Select K virtual machine template weightages as the initial centroids
3: **repeat**
4: Form K clusters by assigning virtual machine template weightages to the closest centroid
5: Re-compute the centroid of each cluster
6: **until** the centroid does not change

3.2. Ranking the host machines based on free CPU and Memory available. Once a VM is chosen, our next step is to find the best host to instantiate the VMs. As already mentioned, MPI based applications spend more time on communication, and the goal is to place VMs such that their communication time will be less. According to studies and experiments, VMs instantiated on the same host with shared memory-based communication will have better network performance than VMs communicating via the traditional TCP/IP

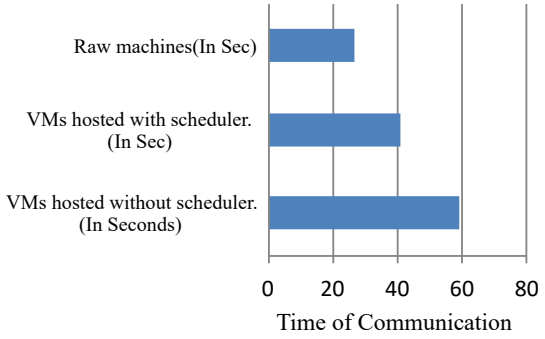

Fig. 3.2: Performance comparison of application hosted on a raw machine, VM with scheduled placement and without scheduling

protocol suite.To use this advantage, our placement strategy places the maximum possible number of VMs on the same host machine with more available resources. In order to find the host machine with the maximum resources, every host machine is ranked based on the available resources. The work considers CPU and memory resources, and hence the host machine that has the most CPU and memory is considered the highest-ranking host machine. Equation 3.2 is used to determine ranking.

$$\begin{aligned} Rank = (x \times percentage - of - CPU - usage - of - single \\ -core - of - host - with - minimum - usage) + \\ (y \times percentage - of - available - memory) \end{aligned} \tag{3.2}$$

where x and y values will range from 0 to 1 depending on the nature of the application the user is requesting. If the application is CPU intensive, then x value will be higher. If it is memory intensive, then y value will be high. A very simple method deployed to set x and y values was to find the time spent by each process in memory and on the CPU by 'ps' command.

3.3. Placement Strategy. Hosting the VM template on a machine is called VM placement. An appropriate scheduling strategy plays a very important role in enhancing the performance of VM. The graph in Fig. 2 shows the performance of the National Parallel Benchmark (NPB), which is an MPI based HPC benchmark on raw machines, VMs hosted without scheduling policy, and VMs hosted with scheduling policy. The experiment was conducted on a private OpenNebula based cloud which uses matchmaking as a scheduling policy. In this policy, the VM will be hosted on a machine with the highest set of resources. We can observe that VMs placed with scheduling show better performance than VMs placed without scheduling. But still, there is a huge gap between the performance of applications on raw machines and VMs.

A placement strategy which considers the characteristics of MPI based HPC applications is proposed in this paper. Communication between individual processes plays an important role in deciding the performance of MPI applications. Communication on VMs is prone to delays because of the virtualization layer. Fig 3.2shows the extra burden an application must bear while communicating through VMs. All the VMs hosted on a machine share the network interface of the host machine, through which they will be connected through a virtual interface. To improve the performance, we conducted experiments using XWAY, which is a shared memory based inter-domain communication mechanism plugin for Xen version 3.0. It provides an accelerated communication path between VMs on the same physical machine by forcing their shared memory based communication.

The graph in Fig 3.3 shows the improvement in shared memory-based communication over normal TCP/IP based communication.

The placement strategy proposed in the work enhances shared memory-based communication amongst VMs which are part of an MPI cluster by placing the maximum possible VMs on a single host. Host machines will

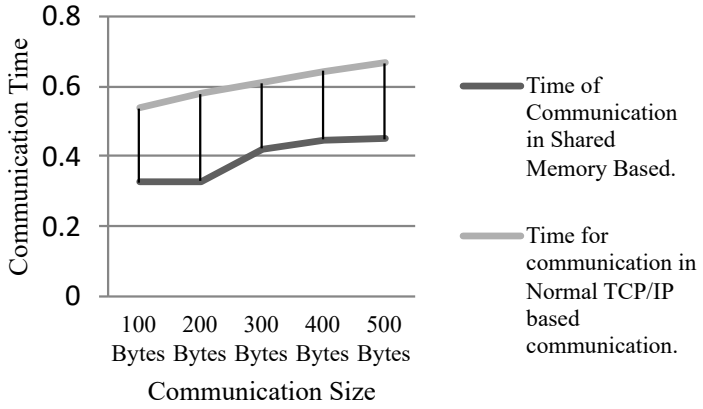

Fig. 3.3: Comparison of shared memory based communication with XWAy to normal communication

be chosen in decreasing order of their rank. If the user is requesting K processes (one VM for one process), then we fit the maximum possible VMs into the highest ranked node machine and continue this procedure in descending order of rank until all ”k” VMs are instantiated.

Algorithm 2 Algorithm for VM Placement

```
Let M ∈ Cluster of VM templates based on similarity metric
R ∈ rank of each host based on resource availability
P = number of VMs requested by user
repeat
    N= Next Highest Rank Node from R
    Fit maximum possible VMs from M in N
until P
```

3.4. Network Bandwidth Reservation. Even after applying the placement strategy, a few communicating VMs may be on different hosts due to a lack of resources on the same host. They use TCP/IP stack-based communication, thereby becoming a bottleneck in the overall performance of the application. A study shows that resource reservation will enhance the performance drastically. But reservation without prior knowledge of application requirements is costly and leads to wastage of resources. The next objective aims at improving the communication performance of VMs hosted on different host machines by adequate reservation of network bandwidth. Open Switch, which is an OpenFlow protocol-based tool, is used for the reservation of bandwidth. This work includes the following two parts:

- Modeling and prediction of network bandwidth resource of an application.
- Reservation of predicted bandwidth using proper tools

In this work, for the given MPI application, we predict the bandwidth requirement depending on the data size communicated between individual processes. Prediction of a variable depends on various known variables. Here, depending on the size of the data transmission and the ideal time of data transmission, we predict the bandwidth. Block diagram Fig 3.1 depicts the proposed model. User requirements in terms of the number and type of VMs and application details are collected. A given application is profiled on a single machine using the MPI profiling tool. Profile data will give us the amount of data communicated by each process. Communication costs are indirectly proportional to the bandwidth available. As the bandwidth increases, communication time reduces. But communication time does not entirely depend on bandwidth. A few other things, like processing delay, buffer availability, also influence the communication cost. For a given data size, if we increase the bandwidth linearly, after a certain limit, communication time remains constant without the

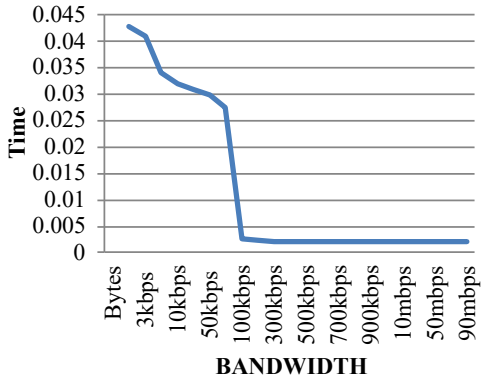

Fig. 3.4: Variation in Time of Data transmission for data size of 10kB for various bandwidths

effect of the increase in bandwidth. Consider the graph shown in Fig.4. For the data size of 10kB, we can see that up to 100kbps of communication time keeps on reducing. However, starting at 100Kbps, time remains constant for all bandwidths. We refer to this bandwidth as "Ideal Bandwidth". For a given data size, any bandwidth below ideal bandwidth decreases the performance, and above ideal bandwidth will result in wastage of resources.

To find Ideal Bandwidth, for the given environment, 'Z' number of bytes will be sent from one VM to another VM with various bandwidths Bi and time taken Ti which will be noted down. Bandwidth B from which time remains constant, is considered as ideal bandwidth. For various sizes of data, Ideal Bandwidth will be noted down and the best fit method will be applied to find the best suited polynomial defining function. Using this function, we predict the bandwidth requirement for the individual processes, which will be reserved for the virtual machines hosting the processes.

4. Experimental Setup and Results. A cloud environment is setup using the Open Nebula cloud platform and the Xen hypervisor is used for VM creation. The Open Nebula is an open-source enterprise-ready cloud management platform that can work with a variety of hypervisors, including those that support a variety of guest operating systems such as Windows and Linux, as well as network and storage support and management tools in a single, tested installable image.

The benchmark application considered for evaluation of our placement strategy is 'NAS Parallel Benchmark' (NPB). It is a small set of programs designed to help evaluate the performance of parallel supercomputers. The benchmarks are derived from computational fluid dynamics (CFD) applications and consist of five kernels and three pseudo-applications in the original "pencil-and-paper" specification. The benchmark suite has been extended to include new benchmarks for unstructured adaptive mesh, parallel I/O, multi-zone applications, and computational grids. Problem sizes in NPB are predefined and indicated as different classes. Reference implementations of NPB are available in commonly used programming models like MPI and OpenMPI.

The original eight benchmarks specified in NPB mimic the computation and data movement in CFD applications. The five kernels are:

- IS → Integer Sort, random memory access
- EP → Embarrassingly Parallel
- CG → Conjugate Gradient, irregular memory access and communication
- MG → Multi-Grid on a sequence of meshes, long- and short-distance communication, memory intensive
- FT → discrete 3D fast Fourier Transform, all-to-all communication

Three pseudo applications are:

- BT → Block Tri-diagonal solver
- SP → Scalar Penta-diagonal solver
- LU → Lower-Upper Gauss-Seidel solver

Table 4.1: Different configurations of Template instances observation

Template	CPU(in %)	Memory (in Mb)	Template weightage
0	25	1024	5.208333
1	10	1024	3.333333
2	20	1536	5.625000
3	10	512	2.291667
4	20	2048	6.666667
5	30	1536	6.875000
6	10	1024	3.333333
7	20	1024	4.583333
8	40	2048	9.166666
9	15	512	2.916667
10	25	2048	7.291667
11	12	512	2.541667
12	15	2560	7.083333
13	10	512	2.291667
14	20	2560	7.708333

Table 4.2: Grouping of templates based on CPU and Memory values

Group1	Group2
Template 0	Template 1
Template 2	Template 3
Template 4	Template 6
Template 5	Template 7
Template 8	Template 9
Template 10	Template 11
Template 12	Template 13
Template 14	

The Benchmark Classes are:

- Class S → small for quick test purposes
- Class W → workstation size (a 90's workstation; now likely too small)
- Classes A, B, C → standard test problems; 4x size increase going from one class to the next
- Classes D, E, F → large test problems; 16x size increase from each of the previous classes

4.1. Grouping the VM Templates. We created 15 different templates, and Table 4.1 shows the different configurations of VM Template instances. Using CPU and memory values, we have calculated the template weightage of each of the templates. Table 4.2 shows the grouping of VM templates based on CPU and memory values. The number of groups is specified by the administrator, and as the number of groups increases, the degree of homogeneity will also increase. When a user requests a VM, the framework will search for a group whose centroid is close to the requested VM criteria.

4.2. Ranking of Host Machines. Three workstations with different workloads are used for hosting the VMs. As a first step, we run the ranking script, and Table 4.3 shows the rank of these workstations.

Values show that workstation named 'Node1' carries high rank with more memory and CPU. Placement policy induced will host the VMs requested by user in 'Node1'. Once the resources are exhausted remaining VMs will be hosted in 'Node2' and finally 'Node3' will be selected if required.

Table 4.3: CPU-MEM-PER values of 3 host machines

Host Machine	CPU-MEM-PER
Node1	44.011
Node2	43.677
Node3	41.879

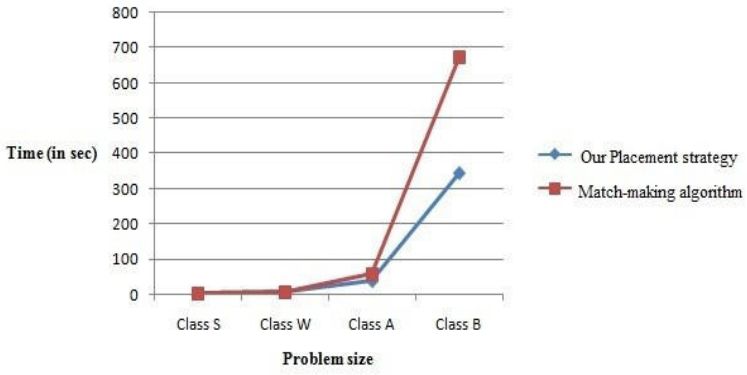

Fig. 4.1: Improved performance in VMs created with our placement strategy and match-making algorithm for four processes

4.3. Network bandwidth prediction and reservation. Individual MPI-based applications have individual processes communicating amongst themselves. This module starts with profiling the application to find the quantity of communication between individual processes. The benchmark application we have considered has all the processes communicating almost an equal quantity of data. If we must handle applications with different data sizes communicated amongst processes, then we can consider the maximum size communicated. Selected VM template and host machines will be modeled, and the bandwidth requirement for a process will be calculated as per the model function chosen by the framework. To test the correctness of this work, a benchmark application was run with various bandwidths amongst VMs carrying processes.

4.4. Comparative study of our Placement Strategy and Open Nebula default match making strategy. OpenNebula uses a default match-making scheduling algorithm. It implements the rank scheduling policy. The goal of this algorithm is to prioritise resources more suitably for the VMs. VMs are placed on the host machine with the most resources in this case.To analyze the performance of our placement strategy, we are using the NAS-NPB parallel benchmark, which is based on MPI. Here we are using the FT benchmark on classes S, W, A, and B. In Class S and Class W communication data sizes are small. But in classes A and B large data sets of around a gigabyte are communicated. The performance of applications over various classes in different placement strategies is compared by varying the number of processes.

4.4.1. Case 1: For 4 processes. Benchmark application execution times for NPB-FT applications using 4 processes are shown in Table IV using our placement strategy and match-making scheduling algorithm. Here, only 4 VMs are formed, and they reside on a single machine, increasing the performance drastically due to shared memory-based communication. We can observe that our placement strategy gives good performance as communication data size increases. On average, we have improved the performance of VMs with our strategy by 552%. In class ‘S’ and class ’W’ as communication size is very small, there is not much difference between the two scheduling strategies. But in classes A and B where maximum communication happens between processes, we can observe that our placement strategy gives very good performance (factor of 7) compared to matchmaking. The graph in Fig 4.1 shows that as communication size increases, our strategy gives good performance.

Table 4.4: Number of Processes: 4

Problem size	Execution time in our Placement strategy (in seconds)	Execution time in Match-making algorithm (in seconds)
Class S	0.47	2.17
Class W	1.17	5.03
Class A	7.91	63.75
Class B	85.08	774.12

Table 4.5: Comparing the results of NPB application obtained from VMs created using Our Placement strategy and Match-making algorithm for 8 processes

Problem size	Execution time in our Placement strategy (in seconds)	Execution time in Match-making algorithm (in seconds)
Class S	2.09	2.65
Class W	5.00	6.64
Class A	38.45	56.18
Class B	342.91	671.77

4.4.2. Case 2: For 8 processes. Table 4.4 shows the execution time of a benchmark application using our strategy and matchmaking strategy for 4 processes.

Table 4.5 shows the execution time of a benchmark application using our strategy and matchmaking strategy for 8 processes. Out of 8 machines, 5 VMs were placed at "Node 1" and 3 at "Node 2". In Class A onwards, data communication will be more, and in such a scenario, our placement strategy gives good performance, and we are observing performance enhancement of 46.11% for Class A and 95.09% for Class B. On an average, we have improved the performance of VMs with our strategy by 70.06% for Class A and B. Fig 4.1 shows the execution time difference between the two methods for 8 process scenarios.

4.4.3. Case 3: For 16 processes. Table 4.6 shows the execution times of two strategies for 16 processes. Once again, we can see good performance from our placement strategy from class A onwards. We can observe a performance enhancement of 36.90% for Class A and 28.62% for Class B. On an average, we have improved the performance of VMs with our strategy by 34.26% for Class A and B. Fig 4.2 shows the execution time difference between the two methods for 16 process scenarios.

In all three cases, we see that for calls S and W, there is no real difference between matchmaking and our strategy. But from call A onwards, we can see the performance enhancement. Class S and W deal with minimum data and other overheads that are greater compared to communication time. But from class A onwards, data communication will be huge, and here our placement strategy works better than others. High Performance Computing application performance depends on communication resources as much as computing resources. Clusters and grids are considered the best platforms for their implementation. The scalability of the cloud will be very suitable for forming better clusters or grids. But at the same time, the heterogeneity and shared resources of the cloud environment pose a challenge.

5. Conclusion and Future work. Even though resources are the main requirement for HPC based applications, the cloud is never considered as a solution for them. There are many cloud vendors providing HPC on the cloud. Still, MPI-based HPC application users are not satisfied with the performance of the application in the cloud. Most of the HPC applications are MPI based, and for them, communication performance plays a leading role as processes need to communicate with each other a lot. VMs hosted on the same machine use shared memory-based communication that leads to improved speed compared to TCP/IP based communication. This work proposes a virtual machine (VM) placement strategy for HPC applications with better communication performance. Our cloud is setup using OpenNebula Cloud management software and the Xen hypervisor for VM creation. We have proposed a placement strategy wherein the maximum VMs of the cluster are hosted on a single machine with maximum resources. Observations show that as communication size increases, our method is providing good results compared to the default placement strategy. In the future, we would like to consider data availability in our placement strategy.

Table 4.6: Comparing the results of NPB application obtained from VMs created using Our Placement strategy and Match-making algorithm for 16 processes

Problem size	Execution time in our Placement strategy (in seconds)	Execution time in Match-making algorithm (in seconds)
Class S	2.44	2.10
Class W	3.99	3.61
Class A	56.03	78.39
Class B	520.24	669.14

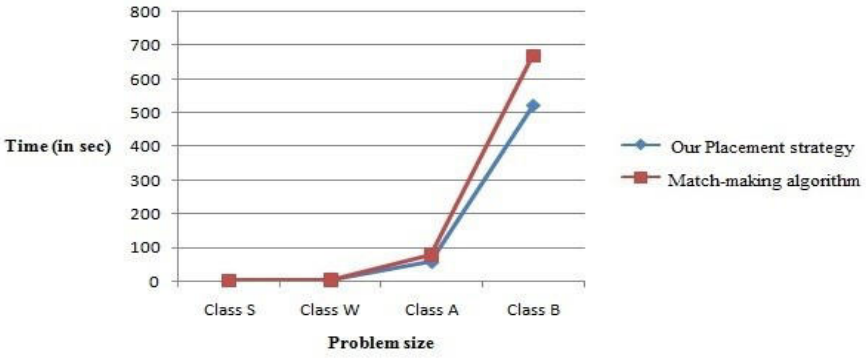

Fig. 4.2: Variation in Time of Data transmission for data size of 10kB for various bandwidths

REFERENCES

[1] Jaliya Ekanayake and Geoffrey Fox, "High Performance Parallel Computing with Clouds and Cloud Technologies" in First International Conference on CloudComp on Cloud Computin Munich, Germany, October 19 - 21, 2009.

[2] Raihan Masud "High Performance Computing with Clouds", http://ix.cs.uoregon.edu/~raihan/HPC_with_Clouds_ Raihan_Masud.pdf, 2011

[3] Danien Warneke and Odej Kao. (2011, February). "Exploiting Dynamic Resource Allocation for Efficient Parallel Data Processing in the Cloud", Journal IEEE Transactions on Parallel and Distributed Systems (TPDS), Volume:22 , Issue: 6), pp 985 - 997.

[4] Hyunjeong Yoo, Cinyoung Hur, Seoyoung Kim, and Yoonhee Kim. (2009 December). "An ontology-based resource allocation service on Science Cloud", International journal of Grid and Distributed Computing, Volume 63 of the series Communications in Computer and Information Science. PP 221-228.

[5] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauery, Ian Pratt, Andrew Warfield, "Xen and the Art of Virtualization", ACM SIGOPS Operating Systems Review, Volume 37(5), Pages 164-177.

[6] Nan Li, Yiming Li. "A study of Inter – Domain communication mechanisms on Xen – based hosting platforms", Course Project Report in course Advanced perating systems, Computer Science Department, UCSB

[7] Rich Wolski, Neil Spring, and Jim Hayes, (1999 October). "The Network Weather Service: A Distributed Resource Performance Forecasting service for meta computing", Journal of Future Generation Computing Systems, 15(5-6):757.768

[8] Shi Na, Liu Xumin, "Research on k-means Clustering Algorithm: An Improved kmeans Clustering Algorithm", "Third International Symposium on Intelligent, Information Technology and Security Informatics", Jinggangshan University, Jian, China, April 2010

[9] Akioka, Sayaka & Muraoka, Yoichi. (2010). HPC Benchmarks on Amazon EC2. 1029-1034. 10.1109/WAINA.2010.166.

[10] Jisha S.Manjaly, Jisha S, (June, 2009). "A Comparative Study on Open-Source cloud Computing frameworks", International Journal of Engineering And Computer Science ISSN:2319-7242 Volume 2, Issue 6. Page No. 2026-2029.

[11] Sempolinski P, Thain D, "Cloud Computing Technology and Science", IEEE Second International Conference on Cloud Computing Technology and Science (CloudMo), Nov. Indiana University, IN, USA, December 2010.

[12] Nicholas Robinson, Thomas Hacker, "Comparison of VM deployment Methods for HPC education", Proceedings of the 1st Annual Conference on Research in Information Technology (RIIT'12), Calgary, AB, Canada, October 2012.

[13] Wei Huang, Matthew J. Koop, Qi Gao, Dhabaleswar K. Panda, "Virtual machine aware Communication Libraries for High Performance Computing", Proceedings of the ACM/IEEE Conference on SuperComputing, Reno, NV, USA, November 2007

[14] Anastassios Nanos, Georgios Goumas, Nectarios Koziris, "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 2010).

Ischia, Italy, August 2010
[15] Richard L. Graham, Galen Shipman, “MPI Support for Multi-core Architectures: Optimized Shared Memory Collectives”, Recent advances in Parallel Virtual machine and Message Passing Interface. 15th European PVM/MPI Users’ Group Meeting, Dublin, Ireland, September 2008
[16] NAS Parallel Benchmarks, http://www.nas.nasa.gov/publications/npb.html
[17] Hai Zhong, Kun Tao, Xuejie Zhang, ”An approach to optimized resource scheduling algorithm for open-source cloud systems” in IEEE transactions 2010.
[18] Hai Zhong, Kun Tao, Xuejie Zhang, ”An approach to optimized resource scheduling algorithm for open-source cloud systems” in IEEE transactions 2010
[19] Zach Hill and Marty Humphrey, A quantitative analysis of High Performance Computing with Amazon’s EC2 Infrastructure: The Death of the local cluster ?,Proceedings of the 10th IEEE/ ACM International Conference on Grid Computing (Grid 2009). Oct 13-15, 2009. Banff, Alberta, Canada. Penguin on demand, http://www.penguincomputing.com/POD/Penguin-On-Demand
[20] Pretto, G.R., Dalmazo, B.L., Marques, J.A. et al. Janus: a framework to boost HPC applications in the cloud based on SDN path provisioning. Cluster Comput 25, 947–964 (2022). https://doi.org/10.1007/s10586-021-03470-6.
[21] Saeed Alshahrani, Waleed Al Shehri, Jameel Almalki, Ahmed M. Alghamdi, Abdullah M. Alammari, ”Accelerating Spark-Based Applications with MPI and OpenACC”, Complexity, vol. 2021, Article ID 9943289, 17 pages, 2021. https://doi.org/10.1155/2021/9943289.
[22] D. J. Milroy et al., ”One Step Closer to Converged Computing: Achieving Scalability with Cloud-Native HPC,” 2022 IEEE/ACM 4th International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC), Dallas, TX, USA, 2022, pp. 57-70, doi: 10.1109/CANOPIE-HPC56864.2022.00011.
[23] A. Fernandez, ”Evaluation of the Performance of Tightly Coupled Parallel Solvers and MPI Communications in IaaS From the Public Cloud,” in IEEE Transactions on Cloud Computing, vol. 10, no. 4, pp. 2613-2622, 1 Oct.-Dec. 2022, doi: 10.1109/TCC.2021.3052844.
[24] İNCE, MUHAMMED NUMAN; GÜNAY, MELİH; and LEDET, JOSEPH (2022) ”Lightweight distributed computing framework for orchestrating high performance computing and big data,” Turkish Journal of Electrical Engineering and Computer Sciences: Vol. 30: No. 4, Article 26. https://doi.org/10.55730/1300-0632.3866.

Edited by: Katarzyna Wasielewska
Received: Aug 4, 2022
Accepted: Apr 4, 2023