# Fine Tuning of Rank Based VM Placement and Scheduling Strategies on Opennebula Based Cloud

**Shreesudha Kembhavi, Shalini Nigam**

***Abstract*: The HPC Clouds are good option over deploying actual physical infrastructure. This helps in running applications efficiently and in cost effective manner as applications leverage resources in pay as you go manner. But HPC clouds suffer performance issues due to Hypervisor layer. This work addresses the issue by coming up with a VM placement strategy considering the intensity of the applications and also resources available in the host machines and avoid performance degradation of parallel running applications. This placement strategy identifies the and ranks the available host machines and places the maximum possible VMs in highest ranking nodes. This avoids communication over the network since the VMs use shared memory for communication.***

***Keywords*: Application Intensity, Cloud Computing, Heterogeneous resources, High Performance Computing, Hypervisor layer.***

## I. INTRODUCTION

High-performance computing (HPC) is the use of parallel processing for running advanced application programs efficiently, reliably and quickly. HPC Applications most of the time use parallel processing t o improve performance. Ideally performance of HPC depends on number of processing elements and data distribution pattern. Cluster and Grid are always considered as the best platform for HPC implementation. Generally owning large cluster/grid is very costly for an organization and may lead to under utilization of resources.

Cloud computing which is also called as utility computing where the user can pay only for what he uses. Infrastructures as a service of cloud provides the resources virtually to the end user, where in the users are hosted with hardware, software, servers etc over the Internet according to the user demands.

The High performance computing users can take the above mentioned advantages of cloud and run their application by making the use of large volume of resources available in the cloud, without deploying the actual hardware infrastructure. many cloud vendors who are providing HPC applications provide static cluster instances which leads to wastage of resources. Hence, HPC on cloud is not widely accepted.

Resources available in cloud like compute, storage and communication resources are shared among VMs which leads to performance degradation.

There are many aspects which decide on the performance of application on cloud platform like placement of VMs, Data locality, Resource Availability etc. This work proposes placement strategy for the virtual machines by ranking the hosts and method for network bandwidth and bandwidth allocation for the virtual machines.

## II. PROPOSED WORK

### A. Profiling of the application

Define Every application spends some time doing CPU related operations and also memory related operations during processing. If the application spends more time in CPU, it is called CPU intensive application and if it needs more memory, then the application is called Memory intensive application.

In this step, we make use of "*ps*" command to get CPU time and memory used by the process and find the CPU and memory utilization of the applications which are further used in ranking of the host machines.

The algorithm to find CPU and memory utilization is as follows:

1. Let X = CPU utilization and Y = memory utilization of the application.
2. X = (Time spent in CPU) ⁄ (Total time spent in CPU and memory usage) * 100.
3. Y = (Memory usage of application) ⁄ (Total time spent in CPU and memory usage) * 100.

Every application is assigned a PID. By making use of this, the time spent in CPU and Memory utilization by the application is realized using "*ps*" command.

### B. Ranking Host

Let us assume there are N hosts added to opennebula cloud for resource utilization.

Consider there are N hosts added to opennebula cloud for resource utilization. We find out Cpu_Mem_percentage of all host machines using the algorithm below.

Manuscript published on January 30, 2020.
***** Correspondence Author

**Shreesudha Kembhavi***, Information Technology, Genba SopanRao College of Engineering, Pune, India. Email: shree.kembhavi25@gmail.com
**Shalini Nigam**, Information Technology, Genba SopanRao College of Engineering, Pune, India. Email:shalini.gour08@gmail.com

Later, the host machines are sorted in descending order of their Cpu_Mem_percentage value and are given ranks. The host machine with highest Cpu_Mem_percentage will be given highest rank and so on.

The algorithm for ranking the host machines is as follows:

1. If N = Number of host machines.
2. For host machines 1 to N,
   //Calculate CPU Memory Percentage of each host using following equation. Substitute X and Y values with the values obtained in the previous step.
   Cpu_Mem_per = ((**X*** minimum CPU usage of single core of the host machine) + (**Y** *((free Memory available in the host machine/Total amount of memory) *100))). (1)
3. Sort the host machines in descending order of their Cpu_Mem_per values and rank them.

### C. Placing of VMs in Highest Ranking Node

The network delay is caused when virtual machines placed in different hosts communicate with each other over the network. Since Virtual Machines placed on the same host machine make use of shared memory. We can avoid the network delay by placing maximum possible Virtual Machines in a single machine.

OpenNebula has the provision to create a pool of VM templates. Template is a file which consists set of attributes that defines a Virtual Machine. Best group of VM template must be chosen by the user based on the requirement of the application and node ranks in an order.

The user must specify the instances of the Virtual Machines as well to run the application. The maximum possible templates must be instantiated in the highest ranking node and remaining instances should be instantiated in further lower ranking node and so on. This leads to lower communication between the Virtual machines placed in different hosts thus lowering the network delay.

The algorithm for the placement is as follows:

```
1. Start
2. Let {C} ϵ Best group of VM template based on the requirement of the application.
3. {R} ϵ rank of each node based on nature of the application.
3. Repeat
    'N'= Next Highest Rank Node from {R}
    Fit maximum possible VMs from {C} in 'N'
5. until {C}= Ø
6. End
```

We will use NAS Parallel Benchmark(NPB) with MPI programming of different classes to analyze the performance after implementation of our ranking strategy and placement of Virtual Machines.

## III. RESULTS AND ANALYSIS

### A. Profiling the application

As mentioned in section II, the host machines are ranked based on the nature of the application. Hence the application is profiled to get the values of x and y used in the host ranking algorithm. This is done making use of "*ps*" command and following values are obtained after the calculations.

$x = 0.1$

$y = 0.9$

### B. Ranking of Host Machines

We are ranking 3 host machines to place maximum number of VMs possible in the Host machines. The priority is given to the host machine which is most suitable to run the application and also considering free CPU and memory usage in the host machines.

**Cpu_Mem_per = ((0.1 * minimum CPU usage of single core of the host machine) + (0.9 * ((free Memory available in the host machine/Total amount of memory) *100)))**

**Table-I: Cpu_mem_per of Host machines.**

| Host machines | Cpu_Mem_per |
|---|---|
| Node | 84.23 |
| Node2 | 83.87 |
| Node3 | 20.77 |

To overcome the performance degradation of the parallel applications, Maximum possible Virtual Machines belonging to an application are placed in a single host machine. This enables Virtual Machines placed in single host to use shared memory for communication.

### C. Performance analysis between our new strategy, old strategy and Match making algorithm

OpenNebula uses by default Match-making scheduling algorithm. We are comparing our results with the default Match making algorithm and as well with old placement [2] strategy which did not consider nature of the application and considering only the free memory and CPU values (i.e., by considering x and y values in equation1 as 0.5) in the host machines.

Here we are using FT benchmark on class S, W, A, B. In Class S and Class W communication data size is small. But in class A and B large data set of around 0.5 G bytes are communicated.

**Table-II: Comparison of Strategies for 4 processes**

| Problem size | Match making strategy | Previous placement strategy | Our new placement strategy |
|---|---|---|---|
| CLASS S | 2.17 | 0.47 | 0.59 |
| CLASS W | 5.03 | 1.17 | 0.88 |
| CLASS A | 63.75 | 7.91 | 7.77 |
| CLASS B | 774.12 | 85 | 71.18 |

Our new strategy has shown the increment in the performance with 72.8% for Class S, 82.5% for Class W, 87.8% for Class A and 90.80% for Class B when compared to Match-making Algorithm. Since communication data size is less, new strategy has shown almost nearest performance to previous placement strategy [2].

There is an increase in performance with 24.78% for Class W, 1.77% for Class A and 15.25 for Class B.

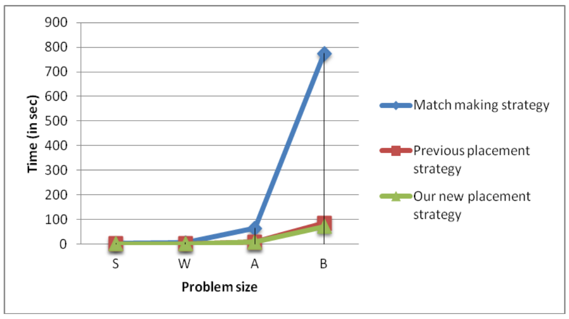

**Fig 1: Performance enhanced in VMs created using Our Placement strategy compared with the Match-making algorithm for 4 processes**

**Table-III: Comparison of Strategies for 16 processes**

| **Problem size** | **Match making strategy** | **Previous placement strategy** | **Our new placement strategy** |
|---|---|---|---|
| **CLASS S** | 2.1 | 2.44 | 2.01 |
| **CLASS W** | 3.61 | 3.99 | 2.95 |
| **CLASS A** | 78.39 | 56.03 | 50.08 |
| **CLASS B** | 669.14 | 520.24 | 474.77 |

New placement strategy shows increase in performance by 4.28% for Class S, 18.28% for Class W, 37% for Class A and 30% for Class B when compared with Match making strategy. Also it shows 17.6% for Class S, 26% for Class W, 10.06% for Class A and 9% for Class B increase in performance when compared with older placement strategy [2].

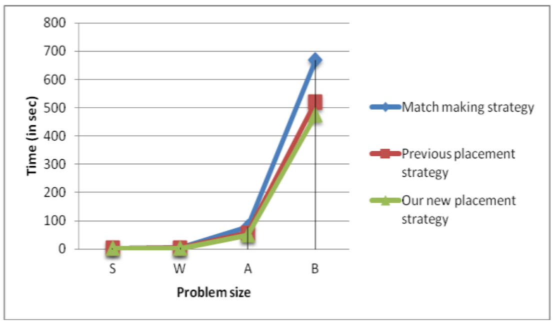

**Fig 2: Performance enhanced in VMs created using Our Placement strategy compared with the Match-making algorithm for 16 processes**

## IV. CONCLUSION

The performance degradation in HPC over cloud is due to resource sharing, hypervisor layer, communication through TCP/IP stack and Bandwidth sharing.

This work proposed Virtual Machine placement strategy by ranking the host machines by considering free CPU and memory as well as the nature of the application. And by placing the maximum possible VMs in highest ranking node performance degradation over the network is avoided since VMs in same host make use of shared memory for communication purpose. Our new placement strategy shows a significant increase in performance for various classes of NAS parallel benchmark in comparison with old placement strategy [2] and OpenNebula's default Match making strategy.

## REFERENCES

1. Ashwini Janagal Padmanabha and Sanjay Harogolige Adimurthy, "Framework for Enhancing the Performance of Data Intensive MPI based HPC applications on Cloud" in Journal of Computer Science, Volume 13, Issue 8, pp 320-328, 2017
2. 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.
3. Jisha S.Manjaly, Jisha S, "A Comparative Study on Open Source cloud Computing frameworks", International Journal of Engineering And Computer Science ISSN:2319-7242 Volume 2, Issue 6 June, 2013. Page No. 2026-2029.
4. Sempolinski P, Thain D, "Cloud Computing Technology and Science (Cloud Computing), 2010 IEEE Second International Conference on Nov. 30 2010 – Dec. 3 2010. Page No. 417-426.
5. Nicholas Robinson, Thomas Hacker, "Comparison of VM deployment Methods for HPC education", RIIT'12 Proceedings of the 1st Annual Conference on Research in Information Technology, 2012. Page No. 43-48.
6. Nan Li, Yiming Li, "A Study of Inter-domain Communication Mechanisms on Xen-based Hosting Platforms", IEEE, 2010.
7. Wei Huang, Matthew J. Koop, Qi Gao, Dhabaleswar K. Panda, "Virtual machine aware Communication Libraries for High Performance Computing", SuperComputing, 2007. SC'07. Proceedings of the 2007 ACM/IEEE Conference on 10-16 Nov. 2007. Page No. 1-12.
8. Anastassios Nanos, Georgios Goumas, Nectarios Koziris, "Exploring I/O Virtualization data paths for MPI applications in a Cluster of VMs: A Networking perspective", Euro-Par 2010 Proceedings of the 2010 Conference on Parallel processing. Page No. 665-671.
9. Richard L. Graham, Galen Shipman, "MPI Support for Multi-core Architectures: Optimized Shared Memory Collectives", Proceedings of the 15th European PVM/MPI Users' Group Meeting on Recent advances in Parallel Virtual machine and Message Passing Interface. Page No. 130-140.
10. Ashwini J P , Nayana M C, Sanjay H A, "Virtual Machine Placement for MPI based Applications on Cloud"
11. OpenNebula.org, www.OpenNebula.org
12. Xen project, http://www.xen.org
13. https://github.com/openvswitch/ovs/blob/master/README.md
14. Octave, http://www.gnu.org/software/octave/
15. NPB,https://www.nas.nasa.gov/publications/npb.html
16. Match making algorithm, http://docs.OpenNebula.org/4.4/administration/references/schg.html#schg-configuration

## AUTHORS PROFILE

**Shreesudha Kembhavi**, have completed BE in CSE and M.Tech in CNE from VTU, Belgaum. She is currently working as Assistant Professor in Department of IT in G S Moze College of Engineering Pune. In 2013, She has completed a project called as VATWI (Visualization and Analysis Tool for weather informatics),a user friendly tool for analysis and visualization of multi-source and multi-scale weather and climate data, which can be used for the efficient and real time analysis of weather and climate studies. as intern in CSIR-4PI Bangalore. Her research area is HPC on cloud during post graduate. She has also attended various workshops on IoT, Machine Learning and Data Analysis.

**Shalini Nigam,** Assistant professor in information technology department of Genba Moze college of engineering, Pune. Graduate and postgraduate in computer science engineering. Her research and publication are in the area of wireless sensor network with mobile sink with big data. The main focus of research is to mitigate with the energy problem of the sensor despite of having large data transfer along with this she has also researched data sciences with cloud. She has published many national and international papers and journals in this area. In addition to this I have attended many workshops on Python and R which is very much essential technologies to learn to work in this area.