Grenze International Journal of Engineering and Technology, June Issue

Distributed Large Memory and Message Passing Interface in Cloud Environment using OpenStack Cloud Platform

Dr. Pranjali P. Deshmukh[1]
[1]Prof. Ram Meghe Institute of Technology & Research Badnera, Amravati, India
Email: pranjali.deshmukh@mitra.ac.in

***Abstract*— Distributed large memory offers the use of large virtual memory by using remote memory distributed over nodes in a cluster. The message passing interface (MPI) plays important role in DLM. MPI-based DLM manages the large virtual memory over the cluster using MPI batch queuing system. In Distributed memory system message MPI is important parallel programming. As the number of virtual machines increases MPI and shared memory parallel programming combined to perform efficient communication in virtual machines [1,2]. Various tasks communicate with each other by sharing memory on the different disk through shared memory distributed computing. If various tasks are trying to access same the same memory location for reading and writing purpose, sometimes it may cause deadlock. In this case synchronization mechanism should handle such situation to keep the integrity of the data [2]. MapReduce and Hadoop are having inbuilt functions to handle distributed shared memory mechanism. The distributed computing with MPI can process thousands of parallel processes in cloud environment**

***Index Terms*— VM, MPI, DLM, Virtual memory.**

I. INTRODUCTION

In the cloud computing load is distributed on various servers to do the load balancing. The term distributed large memory is used where multiprocessor computer system comes. It is another name of multiprocessor system in which each processor is having its own private memory. Each processor is having its own local data resided on its private memory. The computation task is performed on local data. If the remote data is required then process should communicate with other remote processor to access it. There is also term called a shared memory multiprocessor where single memory space is shared by all processors. The processors don't know where the actual data resides. They need to have access permission to avoid the race condition to access shared area. In the distributed system there is some intercommunication, it can be called interprocess communication to communicate amongst processes on each processor [3,4]. The interconnection can be formed using point to point links or separate hardware can be provided. In the cloud computing load is distributed on various servers to do the load balancing. Distributed large memory offers the use of large virtual memory by using remote memory distributed over nodes in a cluster. The message passing interface (MPI) plays important role in DLM. MPI-based DLM manages the large virtual memory over the cluster using MPI batch queuing system. To access this remote memory, researcher

Grenze ID: 01.GIJET.10.2.514_7

work on MPI threads using swap protocol. In experiments, it confirmed that it achieves 493 MB/s and 613 MB/s of remote memory bandwidth with the STREAM benchmark on 2.5 GB/s and 5 GB/s links (Myri-10G x2, x4) high performance of applications with NPB and Himeno benchmarks. Additionally, this system enables users unfamiliar with parallel programming to use a cluster [5].

This paper proposed the Distributed Large Memory System (DLM) which enables user to access the very large memory to compute high performance computation distributed on node servers in cluster format. The results show that DLM using only user-level software achieves better performance than do other low-level remote paging schemes, which typically use a block device for swapping to access remote memory. The stable behavior and higher performance of DLM benefit by DLM's independence of the conventional kernel swapping system. The DLM can tune its parameters independently from the kernel swapping system to obtain higher performance and does not suffer from the unstable behavior of the current kernel swap daemon [67,68]. It also shows that larger page size is more effective to achieve higher performance. The advantages of DLM are not only shown in performance but also in easy availability and high portability, because it is user-level software that needs no special hardware and kernel modification.

The processes in a distributed shared memory model applied to loosely a system are based on the data-passing model**.** Message-passing systems or systems that support remote procedure calls (RPCs) adhere to this model. The data-passing model logically and conveniently extends the underlying communication mechanism of the system; port or mailbox abstractions along with primitives such as Send and Receive are used for interprocess communication. This functionality can also be hidden in language level constructs, as with RPC mechanisms. In either case, distributed processes pass shared information by value. In contrast to the data-passing model, the shared memory model provides processes in a system with shared address space. Application programs can use this space in the same way they use normal local memory. That is, data in the shared space is accessed through Read and Write operations. As a result, applications can pass shared information by reference [6,7]. The shared memory model is natural for distributed computations running on shared memory multiprocessors. For loosely coupled distributed systems, no physically shared memory is available to support such a model. However, a layer of software can provide a shared memory abstraction to the applications [8]. This software layer, which can be implemented either in an operating system kernel or, with proper system kernel support, in runtime library routines, uses the services of an underlying communication system.

II. Distributed Large Memory

The concept of the distributed large memory is to collect and merge all the allocated memory to the virtual machines instances created in a cloud environment. The memory of all VM's is virtually collected are either powered off or shutdown stage. This collected distributed pool of memory is the used to provide small amounts of memory to all the VM who needs them. The Complete process of forming cluster of unused memory and streaming memory from DLM pool. System architecture for DLM is shown in Figure 4.4. Two virtual machines are shown in idle state memory pool of idle machine's memory is created and it provided when running VM in need of more memory.

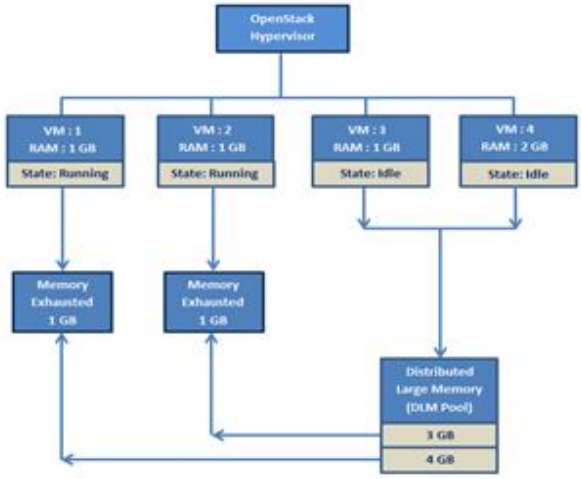

Fig. 1: System Architecture for DLM

A. System Design for Distributed Large Memory (DLM)

Step 1: Check power on virtual machines (VMs)
Step 2: Check idle virtual machines
Step 3: Create memory pool of idle machine's memory if > 2 GB
Step 4: If Required memory of power on VM > Allocated Memory Then:
Step 5: Get Memory from Memory Pool (DLM)
Step 6: VM executes process and task completed
Step 7: If required memory of power on memory < Allocated memory Then:
Step 8: Return memory to free pool
Step 9: Exit

B. Mathematical model for DLM

VM_{AM} =Allocated memory to virtual machines
VM_{RM} =Required memory of power on machine
VM_{Idle}= Number of idle machines
VM_{DLM}= Memory Pool of idle virtual machines if $VM_{AM(Idle)} > 2$ GB
If $VM_{AM(Power\ on)} > VM_{RM}$
Memory in DLM after providing memory to required VM= $VM_{DLM} - (VM_{RM} - VM_{AM(Power\ on)})$

III. Message Passing Interface

MPI or Message Passing Interface is an important component in distributed processing environments such as cloud computing. Through the MPI several processes running in different virtual machines in a cloud platform, can communicate with each other through sending and receiving messages and data required by each other. This concept of distributed message broadcasting makes smooth and seamless operations amongst the VM's.

A. System Design for Message Passing Interface

Step 1: Check power on Virtual machines (VMs)
Step 2: Check idle virtual machine
Step 3: If Required memory of power on machine> Allocated memory Then:
Step 4: Broadcast message to all virtual machines
Step 5: If Required memory < $VM_{Threshold}$ Then:
Step 6: Implement memory ballooning Else
Step 7: Implement Distributed large memory
Step 8: Return the memory to provider VM or Memory Pool
Step 9: Exit.

B. Implementation of Message Passing Interface

MPI or Message Passing Interface is an important component in distributed processing environments such as cloud computing. Through the MPI several processes running in different virtual machines in a cloud platform, can communicate with each other through sending and receiving messages and data required by each other. This concept of distributed message broadcasting makes smooth and seamless operations amongst the VM's. The proposed project work also demonstrates a MPI through a python program shown below.
The mpi4py is python library used for MPI communications.
MPI Codefrom mpi4py import MPI

```
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
if rank == 0:
   data = {'Data' : [7, 2.72,    ,
         'Attributes' : ( 'abc', 'xyz'),
                     'Message' : 'This is MPI message'}
else:
   data = None
data = comm.bcast(data, root=0)
print(data)
```

This shows that two instances can communicates through MPI and messages be sent while performing dynamic memory management. This technique can be used in ballooning as well as in transparent paging. In transparent paging similar memory pages are not duplicated to save memory. Similar pages are shared by virtual machine

C. Experimental Result of Distributed large memory (DLM)

The experimental simulation work shows the VM memory status after implementing DLM. For this simulation work five VMs are launched and allocted memory. Four VMs myubuntu, ubuntu_inst, ubuntu_inst1 and ubuntu_inst2 are in shutdown state and ubuntu is in poweron state. Distributed large memory pool of shutdown VM's memory is created. While creating memory pool minimum memory is kept with shutdown VMs so that any instance of time if VM enters in poweron state then it will need memory. When Poweron VM is in need of memory more than allocated memory it will get memory from DLM pool. As soon as memory requirment goes down it will returned back memory to DLM pool. The memory state of virtual machines and DLM pool before and after is shown in Figures 6.12 & 6.13.

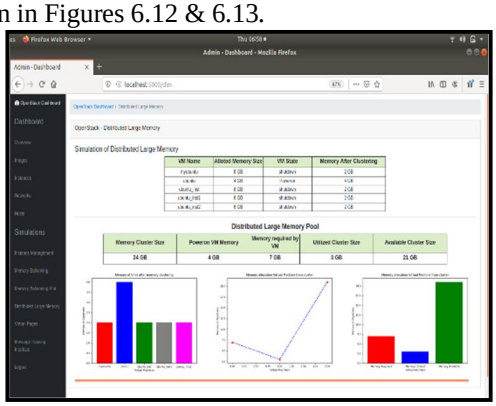

Fig. 2: Screenshot of Distributed Large Memory

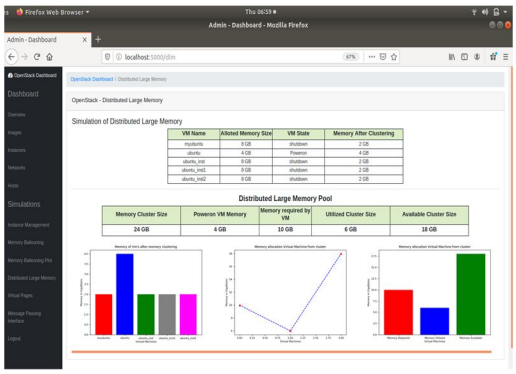

Fig. 3: Screenshot of Memory status of VM after DLM

D. Experimental result of MPI

The experimental simulation shows execution of message passing interface in OpenStack cloud platform. Through the MPI several processes running in different virtual machines in a cloud platform, can communicate with each other through sending and receiving messages and data required by each other. This concept of distributed message broadcasting makes smooth and seamless operations amongst the VM's. The mpi4py is python library used for MPI communications. The MPI is used during ballooning and DLM. When VM is in need of more memory than allocated memory, message will broadcast to all provider VMs. Provider VM in state shutdown will provide the required memory. The message log is stored in database named logger and current MPIs are shown in Figure 6.14.

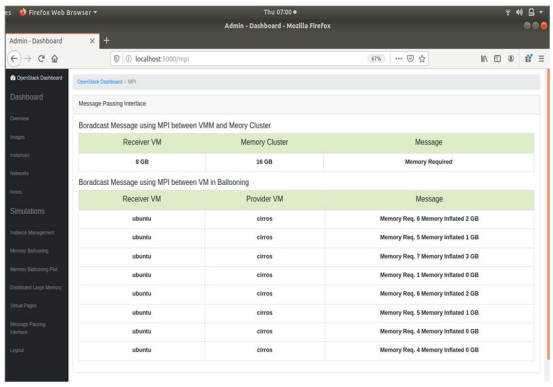

Fig. 4: Screenshot of MPI between VMs and Memory Cluster

IV. CONCLUSIONS

In the cloud computing load is distributed on various servers to do the load balancing. Distributed large memory offers the use of large virtual memory by using remote memory distributed over nodes in a cluster. The message passing interface (MPI) plays important role in DLM. MPI-based DLM manages the large virtual memory over

the cluster using MPI batch queuing system. Distributed large memory is used to form the cluster of unused memory for idle virtual machines and memory streaming is done from large pool to satisfy requirement of VM. The message passing interface can be implemented to make the communication between instances at the time of ballooning and memory streaming.

REFERENCES

[1] Hiroko Midorikawa, Kazuhiro Saito, Mitsuhisa Sato, Taisuke Boku, "Using a Cluster as a Memory Resource: A Fast and Large Virtual Memory on MPI", International IEEE Conference on Cluster Computing and Workshops, CLUSTER 2009.

[2] Hui Zhao Qinghua Zheng, Weizhan Zhang, Yuxuan Chen,"Virtual machine placement based on the VM performance models in cloud", IEEE 34th International Performance Computing and Communications Conference (IPCCC) 2015, pp. 14-16.

[3] P.V.S.S. Gangadhar, Dr. M. Venkateswara Rao, Dr. Ashok Kumar Hota, Dr. V. Venkateswara Rao, " Distributed Memory and CPU Management in Cloud Computing Environment", International Journal of Applied Engineering Research ISSN 0973-4562 Volume:12, No. 24 2017 pp. 15972-15978.

[4] Rakesh Kumar, Bhanu Bhushan Parashar, "Dynamic Resource Allocation and Management Using OpenStack", NCETCE-2014 Supported by: Computer Society Chapter, IEEE Delhi Section ,2014.

[5] Anton Beloglazov , Rajkumar Buyya, "OpenStack Neat: A Framework for Dynamic Consolidation of Virtual Machines in OpenStack Clouds A Blueprint JO", Technical Report CLOUDS-TR-2012-4, Cloud Computing and Distributed Systems Laboratory, The University of Melbourne ER 2012.

[6] A. Husain1, M. H. Zaki2, and S. Islam3, "Performance Evaluation of Private Clouds: OpenStack vs Eucalyptus",International Journal of Distributed and Cloud Computing, 2018 pp.29-36, [Online] Available: .http:// www. publishingindia.com/ijdcc, [Accessed: 2018]

[7] P Aruna, L Yamuna Devi, D Sudha Devi, N Priya, Dr. S.Vasantha and Dr. K.Thilagavathy,"Private Cloud for Organizations: A Implementation using OpenStack ", International Journal of Scientific & Engineering Research, Volume: 4, Issue: 10, October-2013 82 ISSN 2229-5518.

[8] Rakesh Kumar, Neha Gupta, Shilp Charu, Kanishk Jain, Sunil Kumar Jangir "Private Cloud for Organizations: An Implementation using OpenStack" International Journal of Computer Science and Mobile Computing, Volume:3, Issue:5, 2014, pp. 89-98.