International Journal of Research Studies in Science, Engineering and Technology [IJRSSET]
Volume 1, Issue 1, April 2014, PP 87-92

Efficient Load Balancing Using Intelligent Decision on Process & VM Migration

Sharang Telkikar
Department of Computer Engineering
and Information Technology
College of Engineering,
Pune, 411005, India
telkikarsp09.comp@coep.ac.in

Shreyas Talele
Department of Computer Engineering
and Information Technology
College of Engineering,
Pune, 411005, India
talelesr09.comp@coep.ac.in

Siddharth Vanarse
Department of Computer Engineering
and Information Technology
College of Engineering,
Pune, 411005, India
vanarsess09.comp@coep.ac.in

Amit Joshi
Department of Computer Engineering
and Information Technology
College of Engineering,
Pune, 411005, India
adj.comp@coep.ac.in

1. INTRODUCTION

1.1. Load Balancing

It is necessary to understand load to understand Load balancing[1]. Load may be described as number of processes running in queue, load average, CPU utilization, memory utilization, and amount of free CPU time etc. or any combination of the above parameters. Load balancing can be done among interconnected computers in a network or among individual processors in a parallel machine. Load balancing is nothing but the efficient allocation of tasks or jobs to processors for increasing overall processor utilization and throughput.

Actually load balancing is done by VM migration or process migration. But to balance the load it is necessary to measure the load of individual node in network or in a distributed environment. For deciding the load on a node, above mentioned factors in a definition of load are calculated. After calculating the load of node individually, nodes are marked as underloaded/free and overloaded/busy node.

Now in order to balance the load, process or VM is transferred from heavy node to lightly loaded node. In this way load can be balanced in a network of work station or in a distributed environment.

1.2. Process Migration

One of the ways to achieve load balancing in a distributed system is through transferring a process form heavily loaded node to lightly loaded node. It is very useful mechanism for balancing the load on distributed system.

There are two types of process migration[3]. (1)Non-Preemptive Process Migration (2) Preemptive Process migration. Non-preemptive process transfers involve the transfer of processes that have not begun execution and hence do not require the transfer of the process's state. On the other hand, preemptive process transfers involve the transfer of a process that is partially executed. This transfer is an expensive operation as the collection of a process's state (which can be quite large and complex) can be difficult. Typically, a process state consists of a virtual memory image, a process control block, unread I/O buffers and messages, file pointers, timers that have been set, etc. In both types of transfers, information about the environment in which the process will execute must be transferred to the receiving node.

1.3. DMTCP

DMTCP stands for Distributed Multi Threaded Check-Pointing. It is a tool to checkpoint the state of multiple simultaneous applications. It operates directly on the user binary executable, without any Linux kernel modifications. It provides the

Checkpoint-Restart feature which can be used for migration of the process.

2. VIRTUALIZATION

2.1 Hypervisor

The hypervisor is a program that gives the feature to run different virtual machines on single computer. Here they share single hardware host. Each guest system seems to have its own processor and memory. There are two types of hypervisors:

1) Type 1 also called as native hypervisors run directly on the host's hardware with VM resources provided by the hypervisor. This offers higher level of virtualization efficiency. Here guest operating system runs on another level above the hypervisor. This represents the classic implementation of virtual machine architectures. Usually these hypervisors come with security and resource management. Examples of type 1 hypervisor are Microsoft Hyper-V, Citrix Systems XenServer.

2) Type 2 (or hosted) hypervisors run on operating system of the host to provide virtualization services such as memory management, I/O services. These hypervisors are generally used where less efficiency is tolerable generally on client side. KVM and VirtualBox are examples of Type 2 hypervisors.

2.2. Kernel-based Virtual Machine (KVM)

It is a virtualization infrastructure for the Linux kernel. KVM[8] supports native virtualization on processors with hardware virtualization extensions. It is full virtualization solution for Linux hardware having virtualization extension. We can run different virtual machines using OS images. Here each VM has private network card, graphics adaptor and disk. By itself, KVM does not perform any emulation. Instead, a user space program uses the /dev/kvm interface to set up the guest VM's address space, feeds it simulated I/O and maps its video display back onto the host's.

2.3. VM Migration

Another way to balance load in a distributed system is to transfer a VM form heavily loaded node to lightly loaded node. VM migration is simply moving the running VM on a physical machine (source host) to another physical machine (target host) without disturbing any active network connections, while the VM is running on the source host, even after the VM is moved to the target host. It is considered live, since the original VM is running, while the migration is in progress. A guest can be migrated between any hosts. Naturally, a 64-bit guest can only be migrated to a 64-bit host, but a 32-bit guest can be migrated to 32 or 64 bit host.

2.4. Virt-Manager

It is a desktop tool for managing virtual machines. It provides the ability to control the lifecycle of existing machines. Virtual Machine Manager allows users to:

1. Create, edit, start and stop VMs

2. See performance and utilization statistics for each VM

3. View and control of each VM's console

4. Use KVM, Xen or QEMU virtual machines, running either locally or remotely.

5. View all running VMs and hosts and their live performance and resource utilization statistics.

2.5 Network File System (NFS)

It was developed to allow machines to mount a disk partition on a remote machine as if it were a local disk. It allows user to access files over a network as if local storage is accessed. It allows for fast, seamless sharing of files across a network. When VM is moved from one host to another then host write to the same image file.

3. IMPLEMENTATION

3.1. Load estimation and information exchange policy

Traditionally, the load of a node at given time was described simply by CPU queue length. CPU queue length refers to the number of processes which are either executing or waiting to be executed. The processes which are waiting for other system resources are not included. So the CPU queue length does not reflect directly memory utilization. In the proposed algorithm, CPU utilization and memory utilization are used. The system statistics such as CPU utilization and memory utilization of a node changes during the life of processes. For example, the CPU utilization may be high in one second but low in the next second. Therefore it is reasonable to average these statistics over several seconds. In the proposed algorithm, CPU utilization (cpu u) and memory utilization (mem u) are considered as load information parameters to measure load of a node. The following equation is used to calculate each parameter

$$l(avg) = \frac{(p1 + p2 + \ldots + pt)}{t}$$

Where

1. (avg) is the average load metric of the cpu utilization over the previous t seconds for a particular node.
2. p is the information parameter of load. (cpu utilization).
3. p1,…,pt is the value cpu utilization in a previous one second interval.
4. It is the number of time intervals.

3.2. Load classification

The first step in the process transfer determination is to classify the load at each of the nodes[5]. The CPU utilization is divided into three bands: Lightly loaded, moderately loaded and heavily loaded based on the threshold value.

The calculation of the threshold for these parameters is done as follow:

The first step in the process transfer determination is to classify the load at each of the nodes. The CPU utilization is divided into three bands: Lightly loaded, moderately loaded and heavily loaded based on the threshold value.

The calculation of the threshold for these parameters is done as follows:

Calculate load average of each parameter (cpu u) over all nodes. The equation is:

$$l(avg) = \frac{(p1 + p2 + \ldots + pn)}{n}$$

where

1. (avg) is the average load of a given parameter over all nodes.

2. p is the parameter of load cpu utilization

3. l1, …, ln are the current load of cpu utilization of each node derived by load estimation policy

4. n is the number of nodes.

And calculation of moderate band:

The threshold is the average of CPU usage of all hosts. The moderately loaded band is created by adding and subtracting a value, which is the difference between the average and the mean of the maximum and minimum of the CPU usage of the hosts or of 20 percent width, across the threshold, whichever is maximum.

After the calculation of three bands classification is done as diagram:

3.3. Migration Vector Calculation

Now for the nodes that are not in moderate region, load balancing is to be applied .This is achieved by migrating process or VM from heavily loaded node to lightly loaded node so that all the nodes lie in the moderate region. Both the systems try to achieve a moderately loaded band. The value is calculated by taking the difference between the threshold and the current CPU utilization of the lightly loaded node.

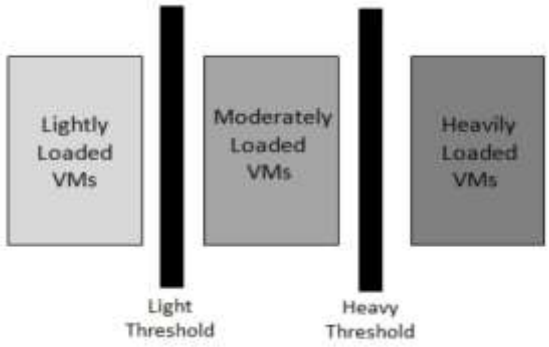

Figure 1. *Each node is assigned one of three regions. Darker region indicates more load on node*

3.4. Migration Policy

After calculating migration vector we need to take into consideration cpu utilization of all the vm's that are running on heavy node and the cpu utilization of those processes that are invoked by DMTCP . Now the process or VM whose cpu utilization is closer to the migration vector calculated in above phase should be migrated.

If the choice is VM then QEMU-KVM's live migration feature is used. The libvirt api virsh command is used for this live migration. And if choice is process then that process is checkpointed and that process is migrated on light node using checkpoint restart feature of DMTCP.

Following flowchart shows the description of policy of program execution

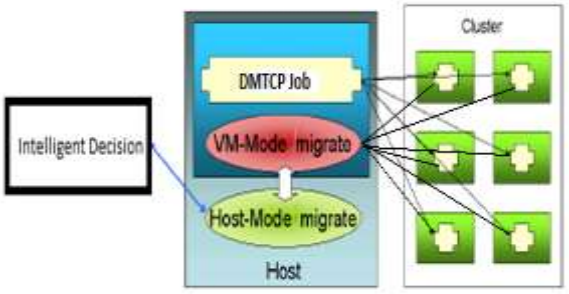

Figure 3. *Architecture containing cluster nodes and central host [4]*

In this architecture, one central host is shown which has both facilities of process migration using DMTCP and also live migration using QEMU-KVM. Thus it is connected to all the other nodes in given cluster

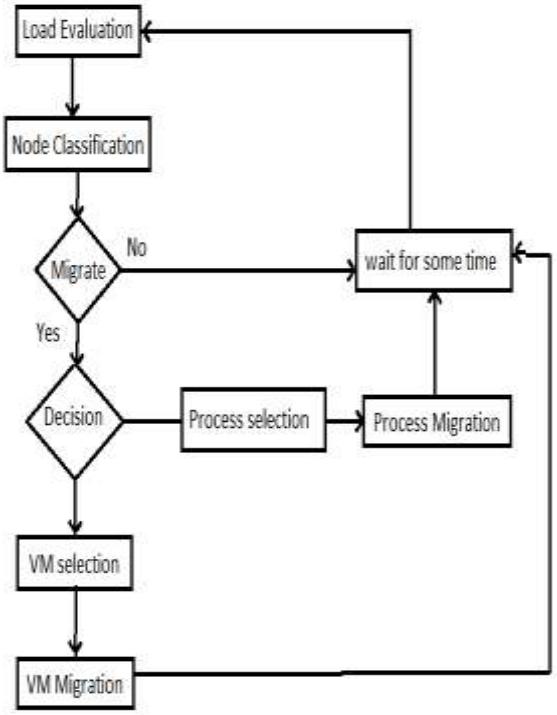

Figure 2. ***Flowchart of different phases of algorithm***

Experimental Setup and Result Analysis

For the assessment of performance of proposed algorithm,

Virtual platform :KVM and storage system: NFS was used. QEMU-KVM and Virtual Machine Manger were installed to manage and migrate VM. DMTCP tool is used to checkpoint the process and migrate it to another node. Three client machines of configuration : Intel Core i5-2400 3.10 GHz * 4 and Memory: 3 GB were used on which libvert, dmtcp, nfs server,qemu-kvm packages were installed.

By applying proposed algorithm we got following results (CPU Utilization in percentage). Table 1,2,3 and 4 shows the nodes and their corresponding CPU utilization after the three iterations of the algorithm. Followed by Fig,4 which shows the variation in CPU utilization of nodes and at the end all lie in moderate region.

Table 1. *State of VMs and Processes on Nodes*

Nodes	VM/Process Name	CPU Utilization on respective Node	CPU Utilization of Node
Node 1	N1V1	51.50	
	N1V2	11.02	46.40
	N1P	3.00	
Node 2	N2V1	47.52	
	N2V2	7.23	21.09
	N2P	20.06	
Node 3	N3V1	96.67	
	N3V2	5.16	63.42
	N3P	25.62	

By using the formulae given in load estimation, load classification, calculation of moderate band, lower and upper limit of moderate band is calculated. Node 1 is in moderate band, Node 2 is in light band and Node 3 is in heavy band. Hence process N3P (according to migration vector calculation) is migrated from Node 3 to node 2 as shown in Table 2.

This process is repeated until all the nodes fall in moderate band. In Table 2, Node 1 is in moderate band, Node 2 is in heavy band and Node 3 is in light band. Hence VM N2V2 (according to migration vector calculation) is migrated from Node 2 to Node 3 as shown in Table 3.

Table 2. ***State of VMs and Processes on Nodes***

Nodes	VM/Process Name	CPU Utilization on respective Node	CPU Utilization of Node

Node 1	N1V1	50.28	
	N1V2	12.32	45.20
	N1P	2.47	
Node 2	N2V1	49.60	
	N2V2	6.59	55.27
	N2P	22.25	
	N3P	23.46	
Node 3	N3V1	96.03	
	N3V2	4.24	28.69

Now in Table 3, Node 1 is in moderate band, Node 2 is in heavy band and Node 3 is in light band. Hence Process N2P is migrated from Node 2 to Node 3 as shown in Table 4.

Table 3. *State of VMs and Processes on Nodes*

Nodes	VM/Process Name	CPU Utilization on respective Node	CPU Utilization of Node
Node 1	N1V1	50.28	
	N1V2	12.32	45.20
	N1P	2.47	
Node 2	N2V1	49.60	
	N2V2	6.59	55.27
	N2P	22.25	
	N3P	23.46	
Node 3	N3V1	96.03	
	N3V2	4.24	28.69

Now in Table 4, all the nodes lie in moderate band and hence load balancing is achieved.

Figure 4 shows that after applying algorithm, dynamic load balancing is achieved as all nodes lie in moderate region.

Now the above depicted algorithm can be applied again after waiting for some fixed time margin.

Table 4. *State of VMs and Processes on Nodes*

Nodes	VM/Process Name	CPU Utilization on respective Node	CPU Utilization of Node
Node 1	N1V1	51.63	
	N1V2	10.62	44.17
	N1P	3.23	
Node 2	N2V1	46.21	
	N3P	31.43	39.73
Node 3	N3V1	95.60	
	N3V2	2.57	42.39
	N2V2	2.31	
	N2P	24.70	

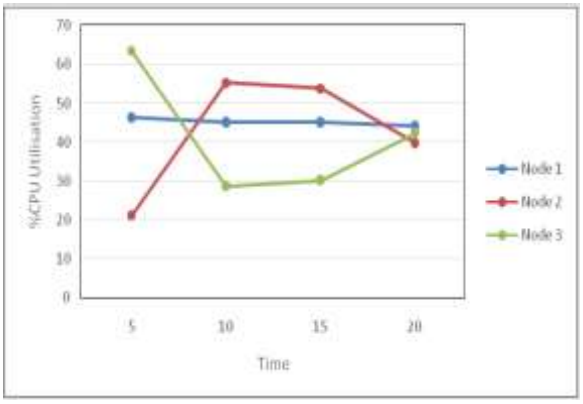

Figure 3. *Variation in CPU utilization of nodes*

4. Conclusion and Future Work

Till date load balancing was achieved either by considering only process migration or VM migration but this proposed algorithm gives an intelligent decision whether to migrate process or VM and dynamically balances the load. Initially all the nodes in the network were imbalanced. After applying the algorithm, all nodes fall in moderate band. Thus load balancing is achieved.

In this algorithm, we migrate VM or processes that are running on node itself. Further load balancing can be improved by migrating processes between different virtual machines running on different nodes. This will improve performance of dynamic load balancing.

Here we mainly focused on CPU utilization parameter, but we can also consider other parameters such as CPU queue length, size of

process, dependency on host and time required for migration.

REFERENCES

[1] Asst. Prof. Vatsal Shah, Asst. Prof. Kanu Patel, "Load Balancing Algorithm by Process Migration in Distributed Operating System," IRACST- International Journal of Computer Science and Information Technology & Security ISSN: 2249-9555 Vol. 2, No.6, December 2012.

[2] Rohan Garg and Komal Sodha and Gene Cooperman," A Generic Checkpoint-Restart Mechanism for Virtual Machines" http://arxiv.org/abs/1212.1787v1.

[3] Vatsal Shah, Viral kapadia, "Load Balancing Algorithm by Process Migration in Distributed Operating System," International Journal of Soft Computing and Engineering (IJSCE), ISSN: 2231-2307, Volume-2, Issue-1, March 2012.

[4] Tal Maoz, Amnon Barak, Lior Amar K. Elissa, "Combining Virtual Machine Migration with Process Migration for HPC on Multi-Clusture Grids," 2008 IEEE International Conference on Cluster Computing 978-1-4244-2640-9/08

[5] Ke Yang, Jianhua Gu,Tianhai Zhao,Guofei Sun, "An Optimized Control Strategy for Load Balancing based on Live Migration of Virtual Machine," 2011 Sixth Annual ChinaGrid Conference, 978-0-7695-4472-4/11, DOI 10.1109/ChinaGrid.2011.28.

[6] Anja Strunk,"Cost of Virtual Machine Live Migration : A Survey," 2012 IEEE Eighth World Congress on Services, 978-0-7695-4756-5/12, DOI 10.1109/SERVICES.2012.23.

[7] Amit Joshi,Akshay Chandak,Krishnakant Jaju, "Dynamic Load Balancing of Virtual Machines using QEMU-KVM" International Journal of Computer Applications (0975 – 8887) Volume 46– No.6, May 2012.

[8] KVM Kernel Based Virtual Machine Red Hat, Inc. 2009