Dynamic Memory Balancing For Resource Management Using Virtual Clusters Machine

S R.Ranjitha[1], R.Ramya,V.Prethivi[2], Mr.G.Arul Selvan ME.,(Ph.D)[3]
[1, 2]Dept of Computer Science And Engineering
[3]Assistant Professor, Dept of Computer Science And Engineering
[1, 2, 3]E.G.S.Pillay Engineering College,
Tamilnadu,India.

Abstract- *Virtualization essentially enables multiple operating systems and applications to run on one physical computer by multiplexing hardware resources. A key motivation for applying virtualization is to improve hardware resource utilization while maintaining reasonable quality of service. However, such a goal cannot be achieved without efficient resource management. Though most physical resources, such as processor cores and I/O devices, are shared among virtual machines using time slicing and can be scheduled flexibly based on priority, allocating an appropriate amount of main memory to virtual machines is more challenging. Different applications have different memory requirements. Even a single application shows varied working set sizes during its execution. An optimal memory management strategy under a virtualized environment thus needs to dynamically adjust memory allocation for each virtual machine, which further requires a prediction model that forecasts its host physical memory needs on the fly. This paper aim to optimize memory control techniques using a balloon driver for server consolidation.*

Keywords- Balloon Driver, Sever Consolidation, Virtualization, Resource Management

I. INTRODUCTION

Recently, virtualization technologies, whose roots can be traced back to the mainframe days, are drawing renewed attention from a variety of application domains such as data centers, web hosting, or even desktop computing, by offering the benefits on security, power-saving, and resource-efficiency[1].Virtual machines (VMs) run on top of the hypervisor or the virtual machine monitor (VMM) 1, which multiplexes the hardware resources. The VMM has the ultimate control on all hardware resources while offering each guest OS an illusion of a raw machine by virtualizing the hardware. No matter if we use a virtualized system for security or hardware multiplexing, we usually boot several virtual machines on a single computer[5].Those machines eventually share or compete for the hardware resources.

In a typical virtualized system, resources, like processors and network interfaces, can be assigned to a virtual machine when needed and given up when there is no demand.The host memory allocation is mostly static–each virtual machine is assigned a fixed amount of host memory in the beginning[17]. Although Xen and VMware provide a ballooning driver to dynamically adjust host memory allocation, existing studies are insufficient to tell when to reallocate and how much memory a virtual machine needs or is willing to give up to maintain the performance of the applications running on it This system proposes xen balloon driver which dynamically monitors the memory usage of each virtual machine, accurately predicts its memory needs,[2]and periodically reallocates host memory to achieve high performance.

In computing, virtual memory (also virtual storage) is a memory management technique that provides an "idealized abstraction of the storage resources that are actually available on a given machine which "creates the illusion to users of a very large (main) memory.

The computer's operating system, using a combination of hardware and software, maps memory addresses used by a program, called virtual addresses, into physical addresses in computer memory[6]. Main storage, as seen by a process or task, appears as a contiguous address space or collection of contiguous segments. The operating system manages virtual address spaces and the assignment of real memory to virtual memory[4]. Address translation hardware in the CPU, often referred to as a memory management unit or MMU, automatically translates virtual addresses to physical addresses[9]. Software within the operating system may extend these capabilities to provide a virtual address space that can exceed the capacity of real memory and thus reference more memory than is physically present in the computer.

The primary benefits of virtual memory include freeing applications from having to manage a shared memory space, increased security due to memory isolation, and being

able to conceptually use more memory than might be physically available, using the technique of paging[12].The virtual memory functions manipulate pages of memory. The functions use the size of a page on the current computer to round off specified sizes and addresses.

The VirtualAlloc function performs one of the following operations:

- Reserves one or more free pages.
- Commits one or more reserved pages.
- Reserves and commits one or more free pages.

You can specify the starting address of the pages to be reserved or committed, or you can allow the system to determine the address. The function rounds the specified address to the appropriate page boundary. Reserved pages are not accessible, but committed pages can be allocated with PAGE_READWRITE, PAGE_READONLY, or PAGE_NOACCESS access[14]. When pages are committed, memory charges are allocated from the overall size of RAM and paging files on disk, but each page is initialized and loaded into physical memory only at the first attempt to read from or write to that page[7].You can use normal pointer references to access memory committed by the Virtual Alloc function.

II. LITERATURE SURVEY

2.1 ENFORCING PERFORMANCE ISOLATION ACROSS VIRTUAL MACHINES IN XEN

Virtual machines (VMs) have recently emerged as the basis for allocating resources in enterprise settings and hosting centers. One benefit of VMs in these environments is the ability to multiplex several operating systems on hardware based on dynamically changing system characteristics[9].However, such multiplexing must often be done while observing per-VM performance guarantees or service level agreements. Thus, one important requirement in this environment is effective performance isolation among VMs. In this paper, we address performance isolation across virtual machines in Xen[1].For instance, while Xen can allocate fixed shares of CPU among competing VMs, it does not currently account for work done on behalf of individual VMs in device drivers. Thus, the behavior of one VM can negatively impact resources available to other VMs even if appropriate per-VM resource limits are in place.

In this paper, we present the design and evaluation of a set of primitives implemented in Xen to address this issue. First, XenMon accurately measures per-VM resource consumption, including work done on behalf of a particular VM in Xen's driver domains[1]. Next, our SEDF-DC scheduler accounts for aggregate VM resource consumption in allocating CPU. Finally, ShareGuard limits the total amount of resources consumed in privileged and driver domains based on administrator-specified limits. Our performance evaluation indicates that our mechanisms effectively enforce performance isolation for a variety of workloads and configurations.

2.2 ADAPTIVE CONTROL OF VIRTUALIZED RESOURCES IN UTILITY COMPUTING ENVIRONMENT

Data centers are often under-utilized due to over-provisioning as well as time-varying resource demands of typical enterprise applications. One approach to increase resource utilization is to consolidate applications in a shared infrastructure using virtualization. Meeting application-level quality of service (QoS) goals becomes a challenge in a consolidated environment as application resource needs differ[5]. Furthermore, for multi-tier applications, the amount of resources needed to achieve their QoS goals might be different at each tier and may also depend on availability of resources in other tiers. In this paper, we develop an adaptive resource control system that dynamically adjusts the resource shares to individual tiers in order to meet application-level QoS goals while achieving high resource utilization in the data center[13]. Our control system is developed using classical control theory, and we used a black-box system modeling approach to overcome the absence of first principle models for complex enterprise applications and systems. To evaluate our controllers, we built a testbed simulating a virtual data center using Xen virtual machines. We experimented with two multi-tier applications in this virtual data center: a twotier implementation of RUBiS, an online auction site, and a two-tier Java implementation of TPC-W[10].Our results indicate that the proposed control system is able to maintain high resource utilization and meets QoS goals in spite of varying resource demands from the applications.

2.3 IMPROVING THE QOS OF WEB APPLICATIONS ACROSS MULTIPLE VIRTUAL MACHINES IN CLOUD COMPUTING ENVIRONMENT

Cloud computing is a hot topic in both industrial and academic areas. Virtualization employed on large data centers forms the basis of cloud computing, which includes CPU, I/O and memory virtualization.[4]Time-sharing of CPU cycles for multiple virtual machines (VMs) has been the main bottleneck of system-level virtualization. How to schedule CPU cycles for multi-VMs to improve the QoS of web applications need

further study. This paper first proposes a CPU management architecture for multi-VMs. Then, we convert the CPU scheduling problem into an integer programming problem. Importantly, we put forward a CPU scheduling algorithm based on utility optimization theory (UOCRS) to increase the global utility[15].Experiments show that our scheme improves the performance of Web applications remarkably.

2.4 MULTIPLE VIRTUAL MACHINES RESOURCE SCHEDULING FOR CLOUD COMPUTING

Cloud computing emerges as a new computing paradigm concerned by both academia and industry[12].Resource management of multiple virtual machines is the core of Infrastructure as a Service. Focusing on the CPU resources, the purpose of this paper is to increase the QoS of web service by properly scheduling the CPU resource across the virtual machines. We formulate the CPU scheduling of multiple virtual machines into an integer programming problem[3].Then, a global regulation algorithm based on utility optimization theory is proposed. Experimental result s show that the regulation system of CPUs can significantly improve the performance of Web applications[11].

2.5LVMM: A LIGHTWEIGHT VIRTUAL MACHINE MEMORY MANAGEMENT ARCHITECTURE FOR VIRTUAL COMPUTING ENVIRONMENT

Virtualization technology recently becomes a hot research topic again in both industry and academics. Some physical hardware such as processors and I/O devices are shared among virtual machines using time slicing, but the memory resource management is relatively complicated[1].Lightweight memory management architecture for multiple virtual machines is proposed, and it includes a mixture of self-adjustment and global-adjustment policies, both of which collaborate with each other to improve the memory efficiency. Experimental results show that our memory management method is effective, and significantly increase the performance of overall system about 20-30%.

III. ARCHITECTURE DIAGRAM

Memory management is the process of controlling and coordinating computer memory, assigning portions called blocks to various running programs to optimize overall system performance. Memory management resides in hardware, in the OS (operating system), and in programs and applications.

In hardware, memory management involves components that physically store data, such as RAM (random access memory) chips, memorycaches**,** and flash-based SSDs (solid-state drives)[8].In the OS, memory management involves theallocation (and constant reallocation) of specific memory blocks to individual programs as user demands change. At the application level, memory management ensures the availability of adequate memory for the objects and data structures of each running program at all times. Application memory management combines two related tasks, known as allocation and recycling[3].

When the program requests a block of memory, a part of the memory manager called the allocator assigns that block to the program.

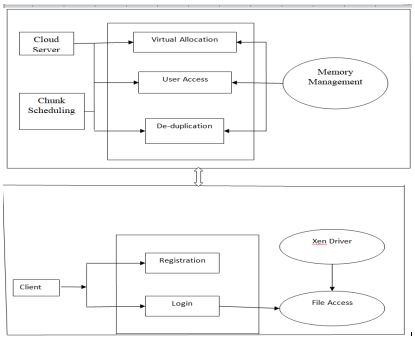

FIGURE 3.1 Architecture diagram

When a program no longer needs the data in previously allocated memory blocks, those blocks become available for reassignment. This task can be done manually (by the programmer) or automatically (by the memory manager).

IV. WORK FLOW

This workflow has levels of DFD design diagrams:

The level-0 has client request for storage to hypervisor,it has a virtual machines, with that help of virtual clusters to allocate memory for the requested devices.

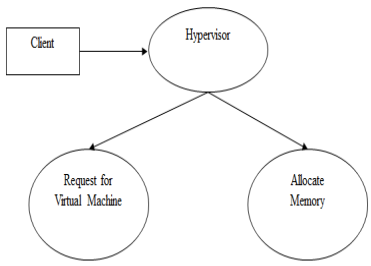

FIGURE 4.1Design Diagram-DFD Level-0

The level-1 constitute of resources and its storage capacity for client who are request for memory it can be

automatically allocated for by our created virtual machines.The client who are not have a sufficient storage to store our files ,in that time it allocate space and dynamically give resources to the particular requested systems.

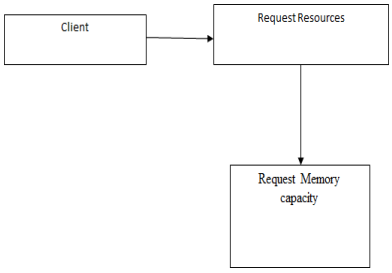

FIGURE 4.2Design Diagram-DFD Level-1

The level-2 contains hypervisor because it has features for automatic allocation of memory and run virtual machines efficiently.so here we can use hypervisor hypervisor to allocate resources for accessing files and storing datas.

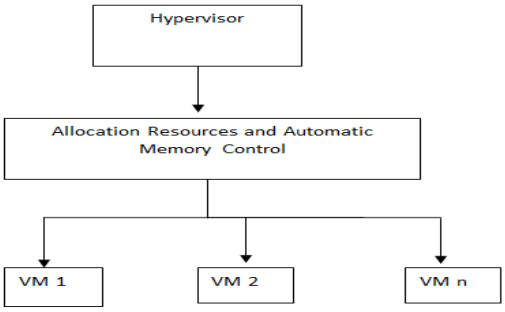

FIGURE 4.3 Design Diagram-DFD Level-2

V. CONCLUSION

Our system aims to optimize the running times of applications in consolidated environments by balancing the memory resources of Xen VMs.We evaluate our optimized solution to memory allocation using real workloads that run across VMs.Our system significantly improves the performances of memory-intensive with multiple virtual machines.This system can be useful for a company or multinational institutes those who want a large resources for huge applications We have to implement this paper with the help of cloud storage acknowledgement, but we were implement as only for a simulation without update in a cloud sever for a simple appliances.

REFERENCES

[1] C.A.WaldspurgerandE.W.William, "Lottery scheduling: Flexible proportional-share resource management," in Proc. 1st USENIX Conf. Oper. Syst. Des. Implementation, 1994.

[2] CarlnA.Waldspurger."MemoryResource ManagementinV MwareESX server".Proceeding of the fifth Symposiumon Operating System design and Implementation, Boston,Dec2002.

[3] The Da CapoB enchmark Suite.(2012). [Online].Available:http://www.dacapobench.org/

[4] The Phoronix Test Suite. (2012).[Online]. Available: http://www. phoronix-test-suite.com/

[5] Strobl,Marius(2013).Virtualizationfor Reliable Embedded Systems. Munich: GRIN PublishingGmbH.pp. 5–6.ISBN 978-3-656-49071-5. Retrieved 2015-03-07

[6] J. Sugerman, G. Venkitachalam, and B.-H. Lim,"VirtualizingI/Odevicesonvmware workstation's hosted virtual machine monitor," in Proc. USENIX Annu. Tech. Conf., General Track, 2001, pp. 1–14.

[7] D. Gupta, S. Lee, M.Vrable,S.Savage, A. C. Snoeren, G. Varghese, G. M. Voelker, and A. Vahdat, "Difference engine: Harnessing memory redundancy in virtual machines,"Commun. ACM, vol. 53, no. 10, pp. 85–93, 2010.

[8] Schopp,D.Hansen,M.Kravetz,H. Takahashi, T. Iwamoto, Y. Goto, H. Kamezawa, M.Tolentino and B.Picco,"Hotplug memory redux," in Proc.Linux Symp., 2005, p. 151.

[9] T.-I. Salomie, G.Alonso,T.Roscoe and K. Elphinstone,"Application level ballooning for efficient server consolidation," in Proc.8th ACM Eur. Conf. Comput. Syst., 2013, pp. 337–350.

[10] C.Clark,K. Fraser, S.Hand,J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in Proc.2nd Conf.Symp.Netw. Syst.Des.Implementation-Volume 2, 2005, pp. 273–286.

[11] W.Zhang,H.He,G.Chen,andJ.Sun, "Multiple virtual machines resource scheduling for cloud computing," Appl. Math. Inf. Sci., vol. 7, no. 5, pp. 2089–2096, 2013.

[12] D. Magenheimer, "Memory overcommit... without the commitment," Xen Summit, pp. 1–3, 2008.

[13] Xen, the powerful opensource industry standard for virtualization.(2013). [Online]. Available: http://www.xenproject.org/

[14] VMware virtualization software for desktops, servers & virtual machines for public and private cloud solutions. (2013). [Online]. Available: http:// www.vmware.com.

[15] KVM. Kernel Based Virtual Machine. (2013). [Online]. Available: http://www.linux-kvm.org/page/Main_Page.

[16] W.Zhang,T.Cheng,H.He and A M.K. Cheng, "LVMM: A lightweight virtual machine memory management architecture for virtual computing environment," in Proc. Int. Conf. Uncertainty Reasoning Knowl. Eng., 2011, vol. 1, pp. 235–238.

[17] Amazon Elastic Computing Cloud (EC2). (2013)[Online].Available:http://aws.amazon.com/ec2.