A SURVEY OF MEMORY MANAGEMENT TECHNIQUES FOR VIRTUAL MACHINES

Sunny Rupabhinda,
Student
Information Technology Department,
Silver Oak College of Engineering & Technology, GTU, Ahmedabad, India

Abstract— Cloud Computing has recently seen explosive growth in the business industry. Ranging from storage requirement to high performance computing, it has been prominently targeting the consumer as well as business market. Cloud computing has been a promising technology in the field of information exchange and flexibility of data. Virtualization has been implemented to exploit the benefits of high performance computing. It eventually enforces the efficient use of resources and leads to overall reduction of cloud data center footprint. However, the memory management has been into research. This paper presents a comprehensive review of several memory management techniques for Virtual Machines.

Keywords— Ballooning, Hot Plug, Memory Management, Memory Reclamation, Page Sharing, Virtual Machine.

I.INTRODUCTION

Virtualization has contributed drastically in reducing the sizes of data centers by maximizing the use of high performance computing. Coupled with cloud computing, virtualization has induced flexibility in executing multiple virtual machines within the same server. Despite the increasing number of servers and exploding growth in the use of mobile computing, the need for all time data availability and access is an undeniable need. In cloud computing, Virtualization helps to serve as an isolated machine which gives an illusion of a physical machine. Virtualization works by multiplexing hardware resources among multiple virtual machines. Even though virtualization offers a plethora of options, it is a challenging task to run an unmodified operating system on a virtualized hardware. This led to development several techniques to modify either the operating system or employing various mechanisms to suit the needs.

Virtual machines require dedicated hardware resources allocated at runtime. In certain cases, these resources tend to be under pressure, leaving the virtual machine in an unsteady state or perhaps, lead to a machine failure. Virtual machine memory is one of the scarce resources which requires coordination and management. Several techniques for memory management are studied in this paper to reclaim memory and dynamically allocate the same.

II.MEMORY VIRTUALIZATION:

A hypervisor – also known as a virtual machine manager works by allocating resources to the Virtual machines (Guest operating systems). Memory virtualization is a technique that implements an address-translation mechanism to virtualize physical memory. VMWare ESX server maintains pmap to translate Physical page numbers to Machine Page Numbers (PPN to MPN).

III.MEMORY OVERCOMMITMENT:

Memory overcommitment is supported by VMWare, which offers an even higher degree of server consolidation than simple single partitioning. Overcommitment refers to the fact that the total allocation of memory is higher than the actual machine memory. Since guest Operating systems do not support dynamic memory changes, it remains constant after the guest is boot up.

IV.MEMORY MANAGEMENT AND RECLAMATION MECHANISMS

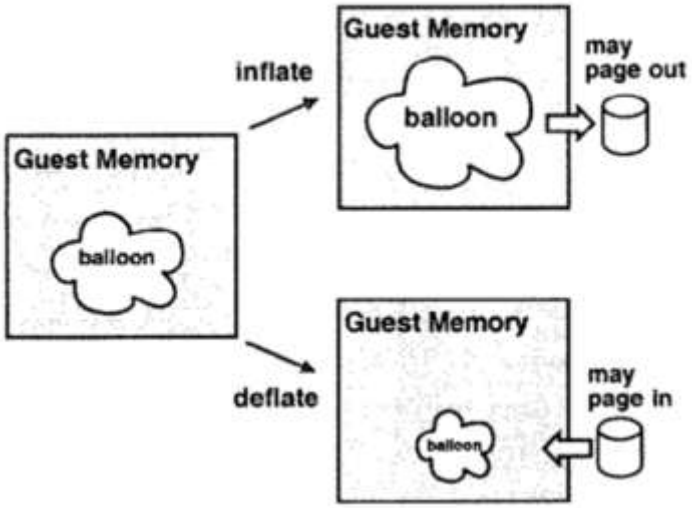
All the systems running under the control of a hypervisor are allocated a fixed memory at boot time. Following memory management mechanisms are proposed:

A. BALLOONING

Ballooning helps by persuading the guest OS to cooperate with the Hypervisor whenever possible. It is done by loading a balloon driver in the Guest OS. An inflation or deflation mechanism is used to reclaim or reallocate memory from or to a Guest OS respectively.

An inflation operation increases memory pressure in the guest OS, which invokes it's native memory management algorithms. The guest OS, upon its decision, decides which pages to reclaim or page them to its own disk. The page numbers are communicated to the hypervisor, which reclaims the machine page. The balloon driver frees up memory by deflation mechanism . [1]

Figure 1 Ballooning



Despite of Ballooning being techniques being used to reduce the memory pressure by reallocation, they tend to badly affect VM performance of applications which manage their own memory. An effective solution to this problem would be to extend ballooning to applications (specifically, Database and Java Runtime), so that the memory can be moved effectively between virtualized instances depending on varying demands. Application Level ballooning re-allocated RAM without shutting down or reconfiguring the applications. [2]

B. PAGE SHARING

In a scenario where a set of VMs may be running instances of same guest OS, having the same components and applications loaded, an exploitation of such opportunities leads to reduces server workloads, thus, supporting higher level of overcommitment.

CONTENT-BASED PAGE SHARING:

Since guest OS cannot be modified**,** a different approach to page sharing employs a mechanism that prevents the need to understand the guest OS code or how the pages were generated. The use of Hashing makes it easier to identify the contents of similar pages with a higher level of efficiency. A hashing table is maintained where a hash value, summarizing a page's contents is a stored. A new hash value is matched against existing values of the table to detect a matching page. A successful match leads to a full comparison of page contents. Upon a successful match, a standard copy-on-write technique can be used to share the pages [1]

TRANSPARENT PAGE SHARING:

A technique for transparent page sharing was introduced by DISCO, aimed to eliminate redundant copies of pages, such as code or read-only data, across virtual machines. Upon identification, multiple pages are mapped to same machine page, marked as copy-on-write. [4]

C. MEMORY HOTPLUG

Memory hotplug aims at expanding system memory on demand keeping in mind the strict requirements of avoiding system downtime.. It takes kernel level modifications to support hotplug and hot-remove.

Memory hot-add comes into picture upon plugging a physical DIMM. The ACPI notifies the OS about he new range of memory address being available. OS kernel leads to initialization of all available memory as free pages. Kernel adds new memory into allocator, making it available for users.

Memory hot-remove, on the other hand, gets memory unavailable for users. It requires page migration before removing the pages from allocator. Page migration leads to performance penalty on machines. [3]

Table 1 HotPlug Phases

State	**Physical**	**Logical**
Function	Communicates the hardware/firmware. Prepares the environment	Changes memory state into available / unavailable for users.

V. CONCLUSION:

In virtualization environments, the demand for resources is application dependent. However, it requires some level of flexibility to meet dynamic user demands by offering on-demand performance. Memory management is a crucial aspect of virtual machine resource management. Dynamic memory allocation mechanisms discussed here are proposed to handle dynamics of memory demands. Even though all of them are effective, the choice of mechanism is highly dependent on application requirement. Although each one has a limitation and a performance penalty during some or the other phase of memory management – appropriate mechanisms have been proposed to make better decisions.

VI. REFERENCES

[1] Waldspurger, C. (2002). Memory resource management in VMware ESX server. ACM SIGOPS Operating Systems Review, 36(SI), p.181.

[2] Salomie, T., Alonso, G., Roscoe, T. and Elphinstone, K. (2013). Application level ballooning for efficient server consolidation. *Proceedings of the 8th ACM European Conference on Computer Systems - EuroSys '13*.

[3] Liu, H., Jin, H., Liao, X., Deng, W., He, B. and Xu, C. (2015). Hotplug or Ballooning: A Comparative Study on Dynamic Memory Management Techniques for Virtual Machines. *IEEE Transactions on Parallel and Distributed Systems*, 26(5), pp.1350-1363.

[4] Edouard Bugnion, Scott Devine, Kinshuk Govil, and Mendel Rosenblum. "Disco: Running Commodity Operating Systems on Scalable Multiprocessors," ACM Trans- actions on Computer Systems, 15(4), November 1997