X-XEN : Huge Page Support in Xen

Aditya Sanjay Gadre
Pune Institute of Computer Technology
adivb2003@gmail.com

Kaustubh Kabra
Pune Institute of Computer Technology
kabrakaustubh@gmail.com

Ashwin Vasani
Pune Institute of Computer Technology
vasani.ashwin@gmail.com

Keshav Darak
Pune Institute of Computer Technology
keshav.darak@gmail.com

Abstract

Huge pages are the memory pages of size 2MB (x86-PAE and x86_64). The number of page walks required for translation from a virtual address to physical 2MB page are reduced as compared to page walks required for translation from a virtual address to physical 4kB page. Also the number of TLB entries per 2MB chunk in memory is reduced by a factor of 512 as compared to 4kB pages. In this way huge pages improve the performance of the applications which perform memory intensive operations. In the context of virtualization, i.e. Xen hypervisor, we propose a design and implementation to support huge pages for paravirtualized guest paging operations.

Our design reserves 2MB pages (MFNs) from the domain's committed memory as per configuration specified before a domain boots. The rest of the memory is continued to be used as 4kB pages. Thus availability of the huge pages is guaranteed and actual physical huge pages can be provided to the paravirtualized domain. This increases the performance of the applications hosted on the guest operating system which require the huge page support. This design solves the problem of availability of 2MB chunk in guest's physical address space (virtualized) as well as the Xen's physical address space which would otherwise may be unavailable due to fragmentation.

1 Introduction

"The Xen hypervisor is a layer of software running directly on computer hardware replacing the operating system thereby allowing the computer hardware to run multiple guest operating systems concurrently. Support for x86, x86-64, Itanium, Power PC, and ARM processors allow the Xen hypervisor to run on a wide variety of computing devices and currently supports Linux, NetBSD, FreeBSD, Solaris, Windows, and other common operating systems as guests running on the hypervisor."[5]

A system running the Xen hypervisor contains three components:

1. *Xen Hypervisor*
2. *Domain 0, the Privileged Domain (Dom0)* – Privileged guest running on the hypervisor with direct hardware access and guest management responsibilities
3. *Domain U, Unprivileged Domain Guests (DomU)* – Unprivileged guests running on the hypervisor.

Types of Virtualization in Xen:

1.1 Paravirtualization

A term used to describe a virtualization technique that allows the operating system to be aware that it is running on a hypervisor instead of base hardware. In this type of virtualization, the operating system is modified in a way that it has direct access to a subset of hardware.

1.2 Hardware Virtual Machine (HVM)

A term used to describe an unmodified operating system that is running in a virtualized environment and is unaware that it is not running directly on the hardware. Special hardware support is required in this type of virtualization i.e. Intel-VT or AMD-V.

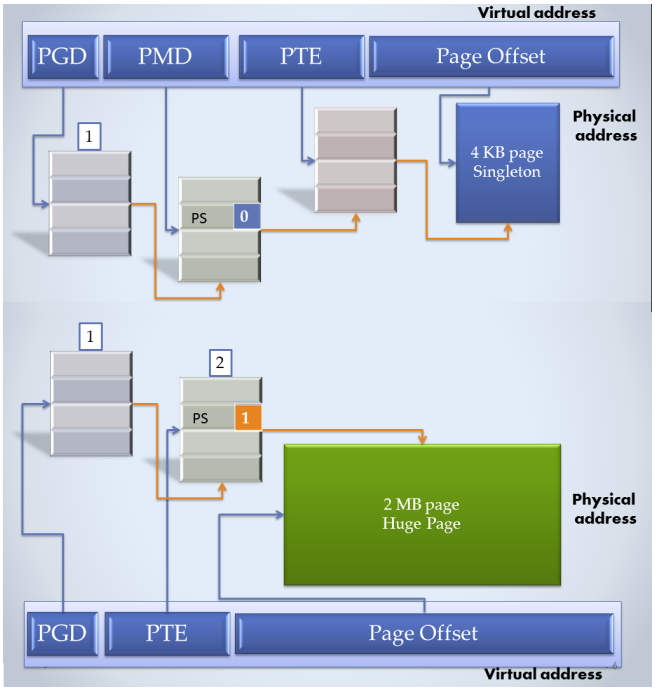

Figure 1: Huge Pages

BIT	-VALUES-			Location
PAE	1	0	0	CR4.5
PSE	X	0	1	CR4.4
PS	0/1	X	0/1	PDE
Page size supported	4K/2M	4K	4K/4M	

Figure 2: Register bits.

2 Huge Pages

Huge pages (Figure 1) are memory pages of size 2MB/4MB depending upon the bits PAE and PSE in CR4 register and PS bit (Figure 2) in the page directory entry.

These bits can be set or cleared using specific instructions in kernel code. The bits table shown above depicts the various page sizes supported according to the bits set or cleared. Huge pages can be reserved in Linux kernel by giving boot time parameter (`hugepages`) option or by executing following command:

```
# echo n > /proc/sys/vm/nr_hugepages
# cat /proc/meminfo | grep Huge
```

3 Problem with huge pages in virtualized environment

When the paravirtualized kernel is compiled with `CONFIG_HUGETLB_PAGE`, support for HugeTLBfs is enabled. Now when the request for hugepage reservation is made to the kernel, it reserves appropriate PFN range.

Problem 1: This contiguous PFN range might not correspond to a contiguous MFN range. When an application allocates a hugepage, the kernel makes the reservation, but when the application tries to access this hugepage, the kernel crashes. The reason for this crash is the MFN corresponding to first PFN of the allocated hugepage range is now treated as contiguous hugepage by the architecture. But the next MFN may not even belong to the domain itself, hence the kernel crashes.

Problem 2: Consider the case when the allocated contiguous PFN range corresponds to contiguous MFN range. Even then the first MFN may not be aligned to a 2MB boundary. In this scenario also the kernel crashes.

Hence, the prerequisites for hugepage allocation in virtualized environment are that allocated hugepages must be contiguous, and must be 2MB aligned in both the kernel address space (virtualized PFNs) and the hypervisor's address space (MFN).

4 Motivation

Huge pages are used by various applications such as JVM, Oracle, MySql for performance improvement. These applications use *mmap* system call for huge page allocation from huge page pool.

As the applications and data grow larger and larger, the need of huge pages (HugeTLB) increases. The database service providers have their databases deployed on dedicated servers where they utilize hugepage support for improvement in performance. When these providers now want to host these dedicated servers on the Xen based cloud, there is a need to provide huge page support in the virtualized environment such that it will be able to utilize the performance improvement of HugeTLB.

5 Earlier work

Solution of this problem was previously attempted by Dave McCraken. In that implementation there is a flag named superpages in the config file specified for DomU. If this particular flag is set to 1 in the config file, then during domain creation all allocations are aligned on 2MB boundaries and in 2MB chunks (i.e. 512 MFNs). Hence, the prerequisite for the domain creation in this case is that hugepages (or 2MB contiguous chunks) equivalent to the amount of domain's memory (memory/2) should be available with Xen's buddy allocator. If these chunks are not available then the domain denies booting. After the domain has booted and begins using memory, then due to pre-existing fragmentation problem the amount of contiguous PFNs required for hugepage allocation may not be available with the domains.

6 Innovation

In the server environment, when the database servers must handle large amount of data and users, they are usually hosted on the dedicated servers. To get the performance benefits they use hugepages instead of normal 4kB pages. When these dedicated servers are to be moved to Xen based cloud environment, that is where our design to solve the problem fits in. In our implementation we allow to specify the number of hugepages that will be required. When the domain is created, the appropriate MFN and PFN ranges for hugepage allocation are reserved. This implementation ensures that once the domain has booted it will always get that specified number of hugepages. The implementation will be further elaborated in Section 7.

7 Implementation

Our implementation works in three phases. In the first phase we reserve the MFNs for hugepage allocation from the Xen's allocator. In the second phase we reserve equivalent PFN range in the kernel for hugepage allocation. The third phase occurs when the request for hugepage reservation for the applications is made by the kernel. These three phases are further elaborated below.

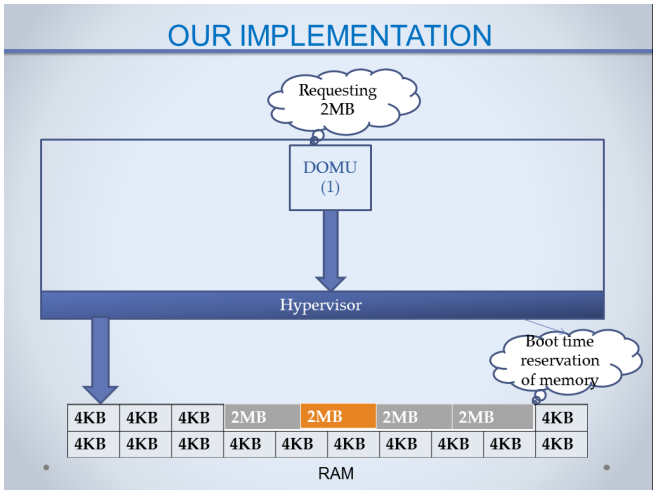

Figure 3: Solution.

7.1 Domain Creation

In our implementation we added a parameter `hugepage_num` which is specified in the config file of the DomU. It is the number of hugepages that the guest is guaranteed to receive when the kernel asks for hugepage using its boot time parameter or reserving after booting (eg. using `echo XX > /proc/sys/vm/nr_hugepages`).

We have introduced two variables in the domain's structure in the Xen hypervisor: `hugepage_num` is an integer which eventually stores the number of hugepages mentioned in the config file of the domain, and `hugepage_list` stores the MFNs of the pages to reserve for the domain. When the domain is being created, the number of hugepages is stored in the variable `hugepage_num`, and then calls to Xen's allocator for 2MB chunks (order 9) are made. The resulting MFNs are added to `hugepage_list`.

```
struct domain {
    :
    //Storing MFN list
    struct page_list_head hugepage_list;
    //Number of hugepages
    unsigned int hugepage_num;
    :
} d;
```

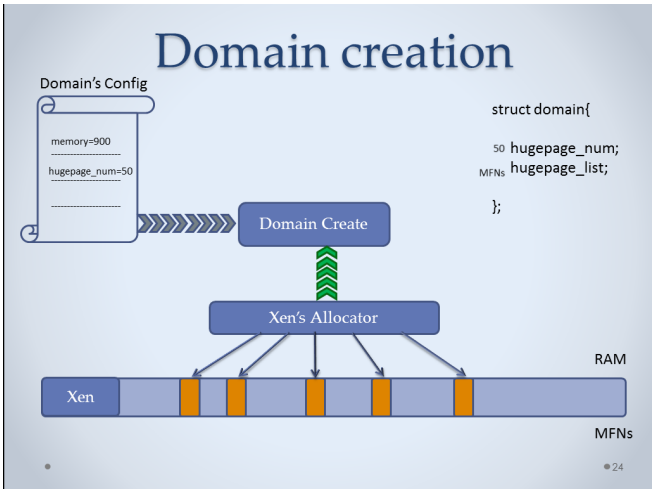

Figure 4: Domain Creation

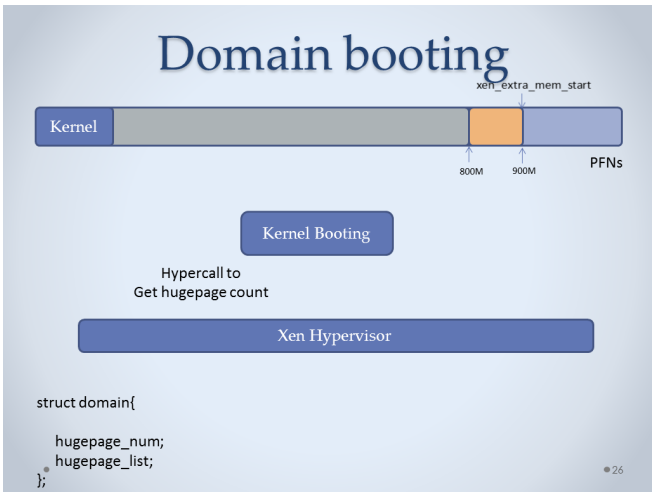

Figure 5: Domain Booting

7.2 Domain Booting

When the domain is booting, the memory seen by the kernel is reduced by the amount required for hugepages. The kernel then makes a hypercall to Xen to get the count of hugepages. Xen takes the count from `hugepage_num` and returns it to the kernel. The kernel now increments `xen_extra_mem_start` by the amount equivalent to the count of hugepages i.e. `count * 2MB`. This is required so that the functioning of ballooning driver is not hampered by our implementation. Now the kernel adds these PFNs (at 2MB intervals) in the `xen_hugepfn_list` which is a newly introduced variable in the kernel. Hence, the PFN range is reserved in the kernel for hugepage allocation.

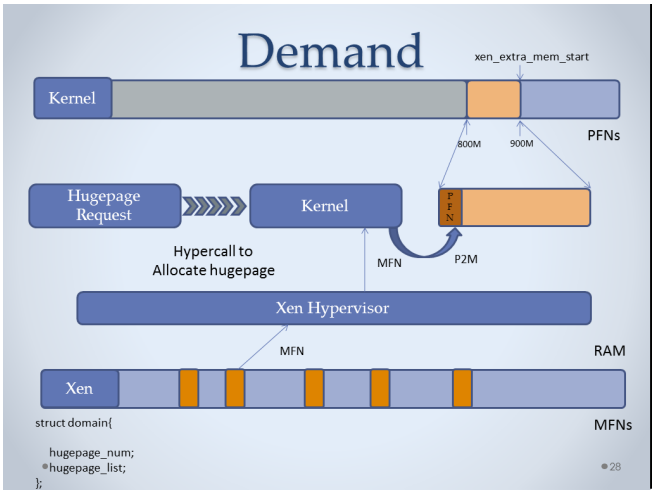

Figure 6: Demand Supply

7.3 Hugepage reservation Request

When a request for hugepage is made by using boot time parameter or reserving after booting (eg. Using `echo XX > /proc/sys/vm/nr_hugepages`), the kernel makes a hypercall to allocate hugepage. The hypervisor then removes one MFN from `hugepage_list` in the domain's structure and returns it to the kernel. The kernel removes one PFN from `xen_hugepfn_list` in the kernel. Then it maps 512 MFNs to the corresponding PFNs beginning from the PFN and MFN just retried. This process is repeated for the number of hugepages requested by the kernel. Hence, all the requirements for hugepage allocation are satisfied i.e. the hugepages are contiguous and 2MB aligned on both kernel (PFN) and hypervisor (MFN) side.

8 Performance

The performance of huge pages was measured by allocating different memory sizes as shown in Figures 7 and 8. Memory was allocated using *mmap* function call and flag `MAP_HUGETLB` was passed in the argument.

9 Constraints

1. Existing live migration techniques need to be modified to support huge page migration.

2. Our implementation may not support 1GB super pages.

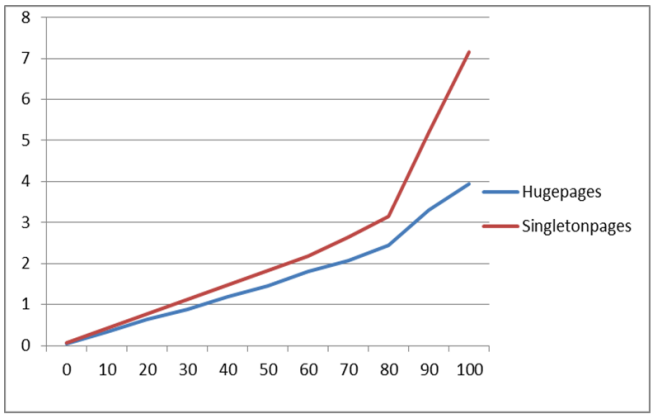

Figure 7: X-axis: Buffer Length & Y-Axis: Time required (Thread =4)

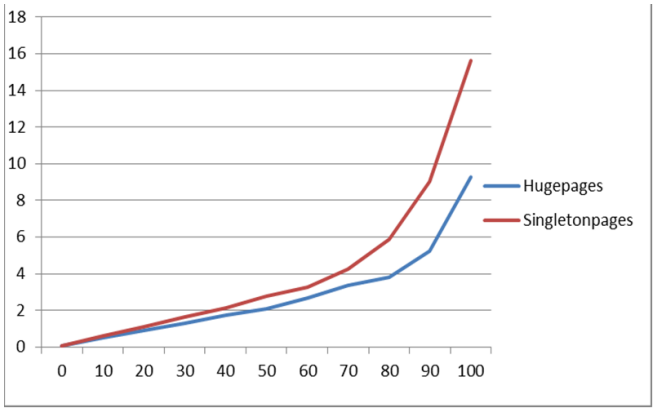

Figure 8: X-axis: Buffer Length & Y-Axis: Time required (Thread =6)

References

[1] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, 2003.

[2] Abhishek Nayani, Mel Gorman & Rodrigo. Memory Management in Linux. http://www.ecsl.cs.sunysb.edu/elibrary/linux/mm/mm.pdf

[3] Tim Deegan, CITRIX Systems. Memory management in (x86) Xen. http://www.slideshare.net/xen_com_mgr/xen-memory-management

[4] Andrés Krapf. XEN Memory Management (Intel IA-32). http://www-sop.inria.fr/everest/personnel/Andres.Krapf/docs/xen-mm.pdf

[5] http://www.xen.org/files/Marketing/WhatisXen.pdf