Ballooning Memory Trap: Dynamic Memory Management in Virtualized Smart TV Environments

Taehun Kim, Junghoon Kim, Keonwoo Kim, Changwoo Min, and Young Ik Eom
College of Information and Communication Engineering, Sungkyunkwan University
{kth1224, myhuni20, kkw0528, multics69*, yieom}@skku.edu*

Abstract

In the near future, smart TV is expected to play a role as a home cloud server with its increased hardware performance. To use the smart TV as a home cloud server, virtualization technique is needed to provide various services for heterogeneous appliances. Of all the virtualulization techniques, the memory virtualization technique is especially important in the embedded environments such as smart TV, which has restricted memory resources. In this paper, we propose a dynamic memory management scheme, called ballooning memory trap, where it dynamically distributes the memory of the smart TV to each virtual machine (VM) by considering the property of the workloads of each VM. We implement our mechanism in the Linux kernel and evaluate the proposed scheme with the representative workloads of smart TV. Our scheme prevents the expensive swap-out operations that cause double-paging problem by retaining the available memory of the host. Furthermore, our scheme guarantees the Quality of Service (QoS) among VMs.

1. Introduction

After Google released smart TV [1, 2] in 2010, many companies such as Samsung, LG, and Sony are participating in development of smart TV. The growth of smart TV gives us various new services (e.g. web browsing, streaming, 3D game, and multimedia) that we have never had before. Recently, smart TV has improved its performance with the support of high-performance hardware (e.g. multi-core processor, 3D acceleration, high-speed network, and various interfaces) and is expected to play a role as a home cloud server. As a home cloud server, smart TV will also provide users the various services through mobile devices. In order to accomplish this, virtualization technology for smart TV can be used to provide a virtual machine (VM) to each user. The hardware trends of the smart TV support this situation. For example, Sony developed a smart TV based on x86 hardware. Also, ARM [3, 4] announced an architectural support for the virtualization [5, 6], which allows the execution of an unmodified guest operating system.

The memory virtualization technique is especially important due to the restrictive memory resources of smart TV in the embedded environment. However, the existing memory management policy [5, 6, 7, 8, 9], which is focused on the managements of desktop or server memory, did not take the properties of the workload of smart TV into account. As a result, the smart TV causes a number of problems. This paper proposes a dynamic memory management scheme for achieving high performance in the virtualized smart TV environment. We firstly analyze features of memory usage in the heterogeneous smart TV workloads. As a result, the VM, which is used for playing multimedia, indiscreetly consumes the memory resource for the page cache. This is a cause of expensive swap-out operation. In addition, this operation may make an additional double-paging [10] problem, which can cause another swap-in and swap-out operations. This problem occurs when the VM is under a memory pressure and tries to swap-out in the same region, which is already swapped out in the host. Experimental results show that our scheme decreases double-paging and swap-out in the host. The remainder of the paper is organized as follows: Section 2 describes the classification and analysis of smart TV workloads. Section 3 describes the memory management problem in virtualized environments and proposes dynamic memory management solution based on analyses of workload. Section 4 presents the evaluation results. Finally, Section 5 draws conclusions and future direction of our research.

2. Analysis of Smart TV Workloads

In this section, we classify the representative workloads in the smart TV and analyze each property of the workload in the native and the virtualization environment. We estimate the memory usage of the each workload. The memory usage is a loaded size in the system. We estimated that size by monitoring the Resident Set Size (RSS). Using the results of these analyses, we propose the ballooning memory trap that guarantees the Quality of Service (QoS) among VMs.

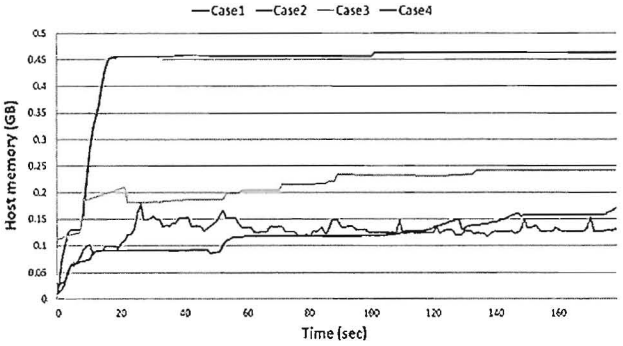

Figure 1. Memory usage of each workload in the native smart TV environment

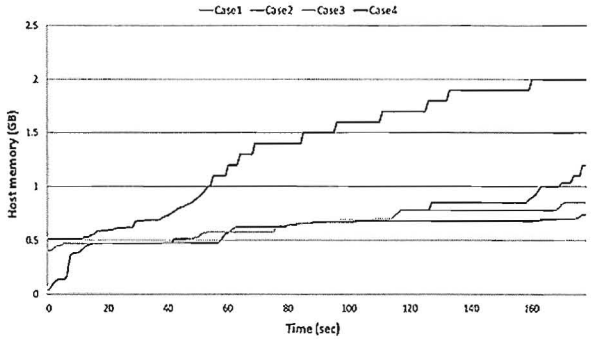

Figure 2. Memory usage of each workload in the virtualized smart TV environment

2.1. Classification of Smart TV Workloads

Table 1. Classification of smart TV workload

Workload Case	Workload Name
Case 1	Internet Web Browsing
Case 2	Game playing
Case 3	Internet Video Streaming
Case 4	Video Playback

Table 1 shows the four representative workloads in the smart TV. Case 1 is an internet web browsing workload which is a basic workload in the smart TV based on the internet. This workload mainly uses the resources of a network. Case 2 is a game playing workload which plays the high quality of 3D game. Case 3 is an internet video streaming workload which plays the internet based video on a real time via YouTube. In addition, case 4 is a video playback workload which plays the high quality movie video of which the resolution is 1920x1080. The video playback workload environment was made using the Xbmc player which is a media center of the Xbox game player made by Microsoft.

2.2. Smart TV Workload Properties

Table 2. Experiment environment

Native Machine	CPU	Intel® Core™ i7 CPU 2.80GHz
	Memory	4GB RAM
	OS	Ubuntu 10.10 (Linux)
Virtual Machine	CPU	Dual Vcpu
	Memory	2GB RAM
	OS	Ubuntu 10.10 (Linux)

The hardware environment for the experiments is shown in table 2. To obtain the accurate result of workload, our experimental environments were configured similar to the environment of the smart TV. Figure 1 shows the memory usage patterns of each workload in the native environment. Case 1 shows a result of the internet web browsing workload. The rate of memory usage is rapidly increased when the users request data or move their pages. Case 2 is a game playing workload. The game playing workload shows the high standards of the memory usage when users play 3D game (e.g. Nexuiz). After loading all the data needed to run the game, the memory is not increased. Case 3 shows a result of the video streaming workload. The rate of the memory usage is rapidly increased when the users click some categories, search keywords or play the videos. Finally, case 4 shows a result of the video playback workload. The memory usage is sharply increased at the beginning of the play, then it keeps about 150MB at the end of the play. Also, the page cache usage is steadily increased until the video is stopped.

Figure 2 shows the memory usage pattern of the VM that runs each workload in the virtualized environments. The cases of Figure 2 are same as the cases of Figure 1. From the case 1 to the case 3, the feature of the workloads is similar to the pattern of the memory usage in the Figure 1. In the case 4, the amount of memory in the VM is increased by using the page cache while the video is playing. Since the VM considers the memory as a real device, the VM uses all the memory as mush as possible. At the end of the video playback, the amount of free memory is increased in the VM. But, the size of the memory in the VM is not decreased.

3. Ballooning Memory Trap

In this section, we analyze the problem which are the semantic gap and the indiscreet page cache usage in the virtualized smart TV environments. We propose the ballooning memory trap based on the experimental result of workloads.

3.1. Semantic Gap and Indiscreetly Page Cache Usage Problem

We run the workload of video playback in the VM. As a result, Figure 3 shows the amount of memory usage on the host and VM. To minimize the access of hard-disk, VM uses the most of memory as a page cache while the video is playing. However, unlike the purpose of using the page cache in an operating system (OS), the page which is less reusable is rapidly increased in the VM.

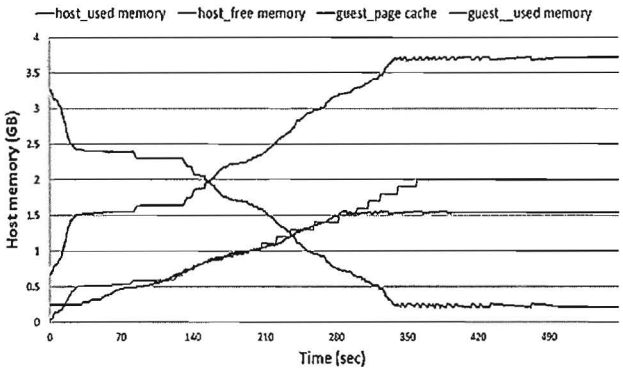

Figure 3. Another view of the same memory caused by the semantic gap between host and guest

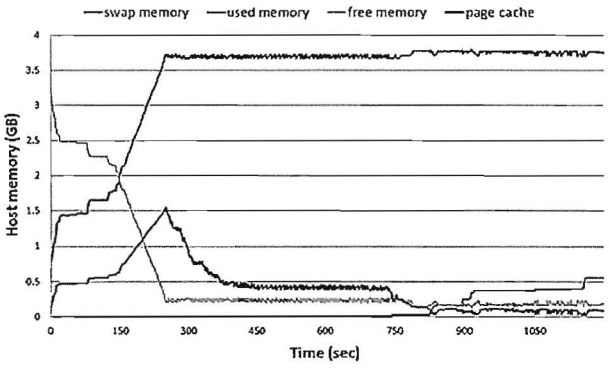

Figure 4. Problem of swap-out caused by the semantic gap between host and guest

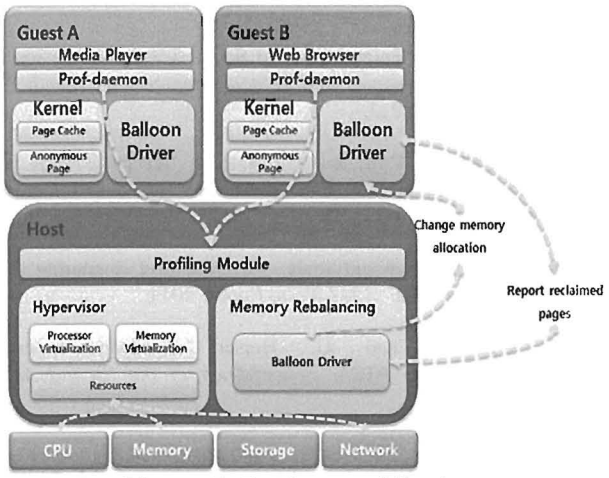

Figure 5. System architecture

It is because page caches grab the more and more page frames, but has never released any of them. As shown in the Figure 4, the swap-out occurs in the host because the lack of the available memory when the VM indiscreetly uses the page cache during the video playback. When the OS lacks its memory, the page frame reclaim algorithm (PFRA) firstly reclaims the page caches which are less reusable. However, the host does not reclaim the free pages and the page which is less reusable in the VM. It is because the host considers all the memory used for heap area or stack area as an anonymous page. As a result, the semantic gap causes the swap-out and the degraded performance of the smart TV system.

3.2. Ballooning Memory Trap Scheme

Figure 5 shows the architecture of our proposed system. Ballooning memory trap consists of three parts: Prof-daemon (PD), Profiling Module (PM), and Memory Rebalancing Module (MRM). PD monitors the memory of the VM and sends that monitored data to the PM of the host. PM receives the memory usage data and helps the MRM to reclaim the memory from the VM. Based on the received data from the PM, the MRM finds the VM which uses a number of free pages and page caches indiscriminately, and

Table 3. Experiment environment

Expression (MB)	Formula	Description
GTM	-	Virtual machine of memory size
RM	RM = GTM/2	Reclaim memory size
MA	MA = RM/2	Monitoring area in virtual machine

reclaims the memory from that the VM. The MRM reclaims the memory of the VM as the following formula.

Table 3 shows a variation of the formula. The GTM is a size of the memory in the VM. The RM is the amount of the memory that the MRM reclaims from the VM. The MA is the area that the PD analyzes the pattern of the memory usage in the VM. Our scheme is operated by the three steps. First, when the host lacks the available memory, the MRM decreases the memory as much as the RM by reclaiming the memory of the VM, and immediately increases the memory as much as the MA to the VM. Second, the PD monitors the MA. Third, the MRM repeats this process if the MA is used the free page and the page caches indiscriminately. VMEXIT [11] which stops the VM is occurred when the VM re-balances the memory. Also, the VMEXIT degrades the performance of the smart TV which has a lower performance than a general home cloud server. The workload of video playback is executed longer than the other workloads. As a result, the VM frequently occurs the VMEXIT for a long time. To minimize the occurrence of the VMEXIT, MRM aggressively reclaims the memory of the VM. The memory requirement can be changed because the VM runs various workloads. The PD monitors the MA to grasp the memory usage patterns in the VM and notify this monitored data to the host. The MRM guarantees the QoS of VMs by decreasing and increasing the memory according to the characteristics of the workload.

4. Evaluation

To solve the problem in the section 3.1 (e.g. semantic gap and the indiscreet page cache usage), we propose the ballooning memory trap.

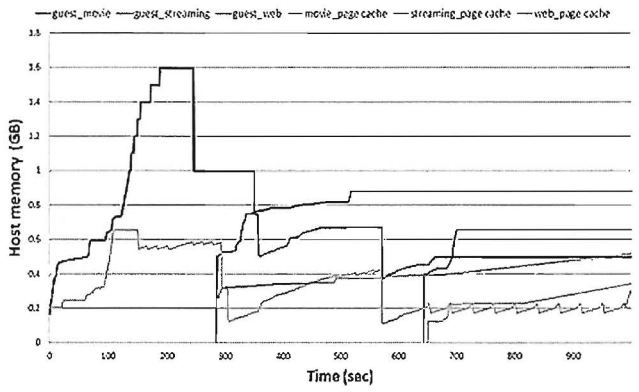

Figure 6. Memory usage estimate among the virtual machines

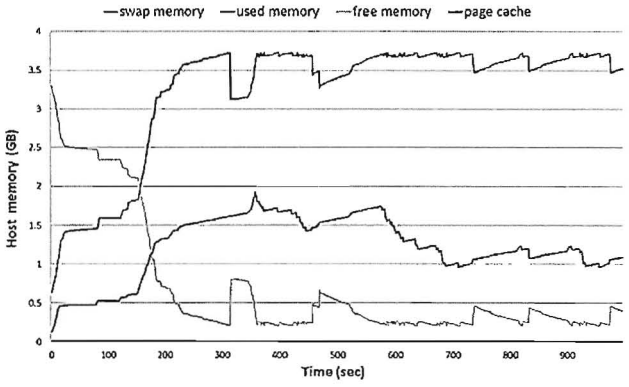

Figure 7. Swap-out prevent in the host

We implement our mechanism in the Linux kernel. In this section, we describe our experimental results. The hardware environment for the experiments is as same as the hardware environment in the section 2.2. The scenario of the experiment is as follows: Our experiment uses three VMs. Each VM is executed by the 300 second intervals and the workload is executed by the order of the video playback, streaming, and the web browser. Figure 6 shows the amount of the memory and the page cache used in each VM. After the VM played the video for the first time, it rapidly uses the page caches and the amount of the using memory is increased. As a result, the amount of free memory is decreased in the host. To guarantee the available memory, the host grasps the pattern of VMs' memory usage by the monitoring tool and reclaims the memory from VMs which indiscreetly use page caches. Figure 7 shows that the swap-out is decreased because the host reclaims the page caches which are less reusable. As a result, we solve the swap-out and the double-paging problem by the ballooning memory trap keeping the quality of video playback.

5. Conclusion

In the virtualized smart TV, the expensive swap-out is occurred by the semantic gap problem. The swap-out degrades the performance of the systems greatly. To solve this semantic gap problem, we analyzed the representative workloads in the smart TV systems. Also, we proposed the ballooning memory trap scheme that reflects the feature of the workload in the virtualized smart TV system. Our scheme monitors the memory usage in VMs and dynamically balances the size of the memory. To evaluate the proposed scheme, we implemented our scheme in the Linux kernel. The results of experiment show that our scheme can reduce the swap-out operations of the host caused by the memory semantic gap. In the future work, we will consider not only the other workloads but also the more VMs.

※Acknowledgment

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (2012-0006423)

6. References

[1] H. R. Choi. The Potentialities of Social TV for Expansion of Smart TV. *Electronics and Telecommunications Trends* 33, pages 607-616, 2012.

[2] K. S. Cho, W. Ryu, and H. W. Lee. Service Trends and Prospect on Smart TV. *Electronics and Telecommunications Trends26,* pages 1-13, 2011.

[3] R. Mijat and A. Nightingale. Virtualization is coming to a platform near you. *ARM White Paper,* 2011.

[4] P. Varanasi and G. Heiser. Hardware-supported virtualization on ARM. *Proceedings of the Second Asia-Pacific,* 2011.

[5] R. Micael and G. Kartik. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. *VEE '09 Proceedings of the 2009 ACM SIGPLAN/SIGOPS,* 2009.

[6] A. Carl. Memory resource management in VMware ESX server. *ACM SIGOPS Operating Systems Review OSDI '02,* 2002.

[7] A. Koto, H. Yamada, K. Ohmura, and K. Kono. Towards Unobtrusive VM Live Migration for Cloud Computing Platforms. *APSYS 2012,* 2012.

[8] D. Williams, H. Jamjoom, and YH. Liu. Overdriver: Handling memory overload in an oversubscribed cloud. *ACM SIGPLAN,* 2011.

[9] Y. Zhang, A. Bestavros, M. Guirguis, and I. Matta. Friendly virtual machines: leveraging a feedback-control model for application adaptation. *Conference on Virtual.* 2005.

[10] R.P. Goldberg and R. Hassinger. The double paging anomaly. *Proceedings of the National Computer Conference and Exposition.* Page 195-199, 1974.

[11] A. Gordon, N. Har'El, A. Landau, and M. Ben-Yehuda. Towards exitless and efficient paravirtual I/O. *SYSTOR '12 Proceedings of the 5th Annual International Systems and Storage Conference,* 2012.