Working Set Estimation

Background

The VMs represent various applications or services, each with varying levels of importance. The challenge lies in allocating memory fairly and efficiently.

Traditionally, the system allocates memory based on overall usage, similar to splitting rent equally among roommates. However, this approach has limitations:

- **Focus on Usage, Not Importance:** A high-usage VM might not be critical, while a low-usage VM might be essential but simply require more memory for optimal performance.
- **Inefficient Allocation:** Less important VMs could occupy memory that would be more beneficial for critical ones.

Problem Description

Virtual machines (VMs) often face challenges in memory management, especially in environments where a physical machine (PM) hosts multiple VMs simultaneously. This situation can make it difficult to allocate memory efficiently. When VMs are provided with static memory allocations, there is a risk of underutilization because some VMs may not fully utilize the allocated memory. On the other hand, if memory allocation is not static, VMs with lower priority may consume more memory than necessary, leading to underutilization and inefficiency.

To address these issues, it's crucial to monitor the working set of each VM. The working set represents the actively used memory by a VM. By analyzing the working sets, administrators can identify VMs that have excess memory and reallocate that memory to VMs that need it more urgently. This approach optimizes memory usage across the PM, ensuring that resources are efficiently utilized without compromising performance or causing underutilization issues.

Our Solution: Working Set Estimation

To address this challenge, we propose a solution based on **working set estimation**. This approach focuses on understanding the actual memory needs of each VM, not just its allocated memory.

By estimating the working set, we can:

- **Prioritize Critical VMs:** Ensure essential VMs have the resources they need first.
- **Optimize Remaining Memory:** Allocate remaining memory more efficiently based on the potential benefit for each VM.

Working set estimation allows us to move beyond a simplistic allocation system and establish a fair and efficient approach thaIt prioritizes critical services while maximizing resource utilization.

Problem Solution Detail

We tried using the approach as mentioned in the memory management paper. So we tried to invalidate the TLB entries and MMU entries but that did not work. We also tried to track the working set by marking the ACCESSED and PRESENT as absent and then on a trap from the VM marking it as present. But again, this idea also didn't work out.

So we tried to track the dirty page by using a **ioctl** KVM API - KVM_GET_DIRTY_LOG. KVM. KVM_GET_DIRTY_LOG tracks the number of pages dirtied by the VM since the last call to this **ioctl** API. Through this we aim to track the working set of the running VM. We tried setting up the the page table for a 2MB and 1GB VM. The page table entries for the third and last level are filled according to the size of the VM.

Experimental evaluation

We tested using four different scenarios of synthetic workloads by simulating a code similar to matrix multiplication for two arrays in two scenarios. Other workloads are accessing locations in the VM random number of times, whose value is given by the random number generator. The sizes for the arrays are set according to the the size of the VM otherwise it doesn't run and the random number generator generates a value from 0-150. Also the two arrays are of **int** and **unsigned long long** types(ULL).

Figures 1 and 2 represent the working set for a 2MB and 1GB VM respectively. The four workloads are combinations of different types of accesses -

(1) Two arrays setup(int type and ULL type) and a random loop for accessing pages **[VM1]**
(2) Two arrays setup(int type and ULL type) **[VM2]**
(3) One array setup(int type) **[VM3]**
(4) One array setup(ULL type) **[VM4]**

The corresponding dirty pages tracked are after two calls to a hypercall(HC_numExitsByType) in the guest code .

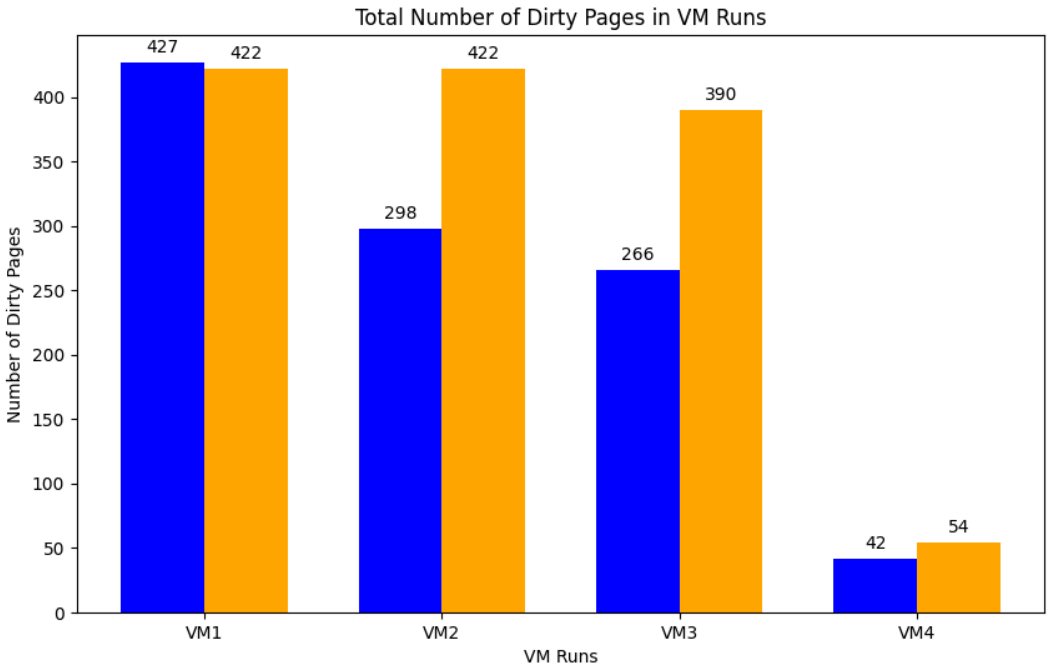

Figure 1. VM working set for a 2MB VM(512 pages)

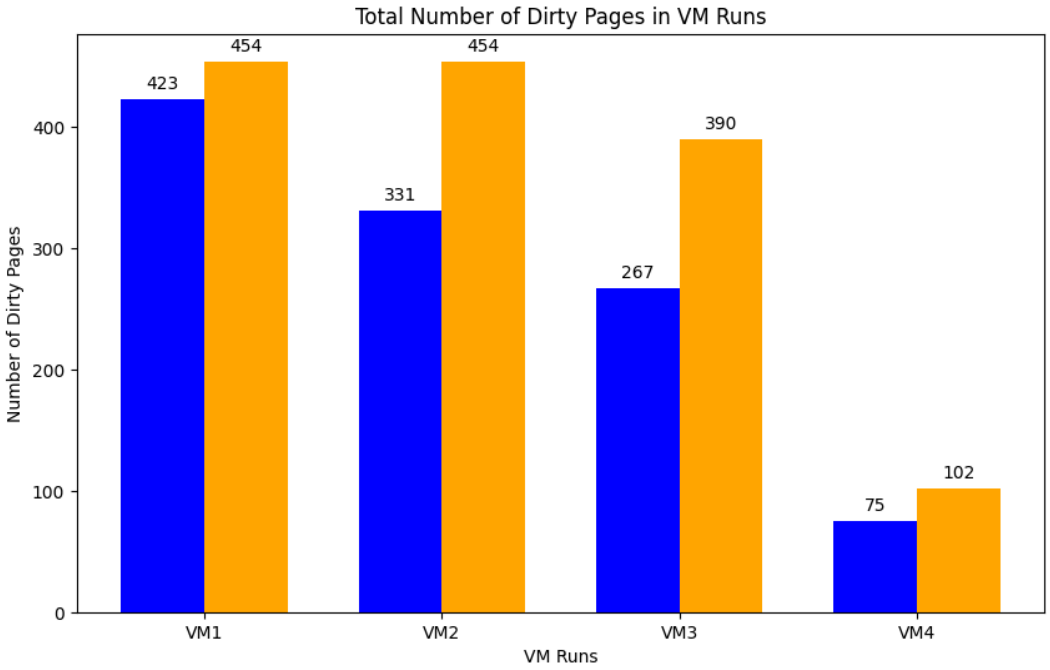

Figure 2. VM working set for a 1GB VM(262144 pages)

Discussion

We observe that this technique is not very effective as this technique tracks only dirty pages to estimate the working set. Also, currently this technique is only tracking dirty pages and not the pages that are just read by the VM. Even if it tracks the read pages, it could only track the maximum working set size as the number of pages allocated to the VM.

A better technique could be to invalidate the TLB and MMU entries and then track the accessed entries but that is also costly as it would lead to page faults.