CS730 project
Empirical analysis of dirty tracking overhead: COW vs Dirty Bit

Tabish Ahmad 21111060
Nikhil Molugu

April 2022

1 Introduction

Finding amount of memory change is a task crucial for applications like CRIU. The project aims to analyse the overheads due to dirty tracking of two well known techniques - COW technique and PTE dirty bit technique.
We implement the technique in this project. On some benchmark program(s), check the overhead due to them.

2 Design

2.1 Dirty Bit

In this technique, we check for dirty bit in all the valid PTEs of the process. First, clear dirty bits, then after some time, scan all valid PTE's dirty bits. The PTEs with dirty bit set means that this memory area was changed.

3 Implementation

3.1 Dirty Bit

We wrote a driver which upon insertion creates a character device of name pgdev. Inside the ioctl function on the char device, it traverses the page table and clears the dirty bit of all the PTE entries. So, to track an applications dirty bits, say bench.c we have to modify it slightly to add signal handlers. In those handlers are the invoking of clearing dirty bits system call and setting dirty bits system call. We wrote a script which will send the signals on respective times to bench.c to invoke these calls. The log of virtual address which have been written into is in the file dirty

3.2 CoW approach

In this approach we modified the fault.c. We added a hook into the page fault handler of linux. First at the beging of the tracking we clear all the R/W bits of all PTEs using our char device's ioctl call, invoked by sending a signal. As, we modified the page fault handler and we made every PTE entry to read, every write access will cause a page fault. Inside the page fault out hook gets called. Inside our hook we are making our PTE entry to R/W.

4 Results

Benchmark Program: Allocate some memory, clear dirty bits (using kernel module) make changes to the memory, scan dirty bits(again using kernel module).

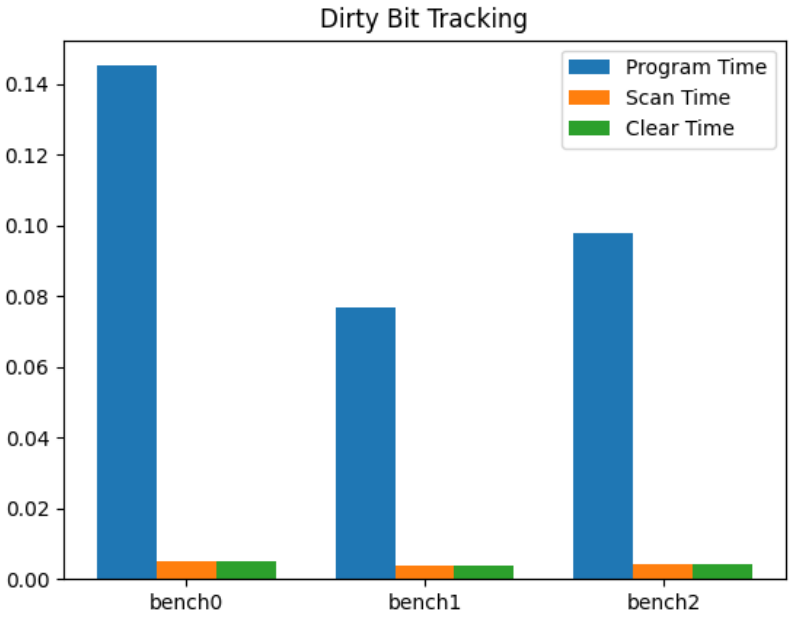

Figure 1: Overheads from dirty tracking

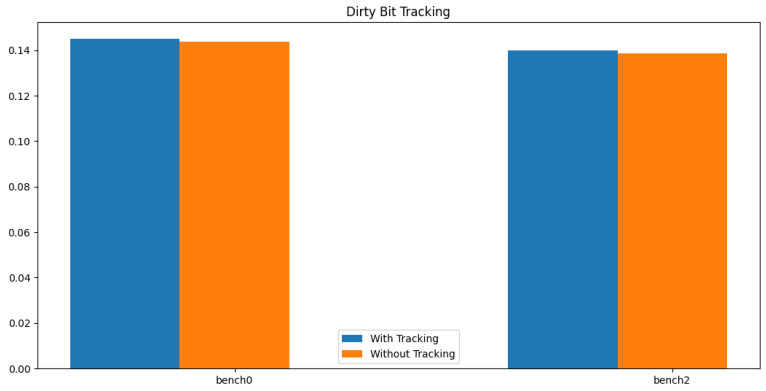

Figure 2: With vs Without Dirty Tracking

5 Conclusion

We were able to implement dirty bit tracking method.
We show the overheads introduced by dirty bit clearing and checking.