COL 331 Major

Viraj Agashe

TOTAL POINTS

73 / 80

QUESTION 1

Q1 10 pts

1.1 (a) **3 / 3**

+ 0 pts Incorrect/Not-attempted

✓ **+ 2 pts** *High-level idea*

✓ **+ 1 pts** *Details - Circular linked list / mm_struct*

1.2 (b) **3 / 3**

+ 0 pts Wrong/Not attempted

✓ **+ 1.5 pts** *PMD + young page*

✓ **+ 1.5 pts** *Details/Datastructure/False +*

1.3 (c) **4 / 4**

+ 0 pts Incorrect/Not attempted

✓ **+ 2 pts** *High level details (indicator of recency etc.)*

✓ **+ 2 pts** *Proper Explanation (slide 61) / LRU overheads*

QUESTION 2

2 Q2 **10 / 10**

✓ **- 0 pts** *Correct*

- 6 pts Correct Part 1

- 4 pts Correct part 2

- 3 pts Part 2 done partially correct/Use case not given ie what situation/application would lead to system preferring one over the other. Note please consider this before applying for regrade

- 2 pts Haven't specified conditions based on swappiness value/Done it incorrectly. See memory-systems ppt slide 68

- 10 pts Incorrect/Not attempted

- 8 pts Right direction needs explanation

QUESTION 3

3 Q3 **8 / 10**

+ 10 pts Correct

+ 5 pts Partially correct

+ 0 pts Incorrect/ Not attempted

+ 8 *Point adjustment*

QUESTION 4

Q4 10 pts

4.1 (a) **2 / 4**

+ 4 pts Same performance

+ 0 pts Incorrect

✓ **+ 2 pts** *Correct answer but with an incorrect explanation or wrong diagram.*

1 Wrong diagram

4.2 (b) **6 / 6**

✓ **+ 6 pts** *RAID 10 is more reliable.*

+ 0 pts Incorrect/Unattempted

+ 3 pts Correct answer with incorrect explanation or wrong diagram

QUESTION 5

5 Q5 **10 / 10**

**#1- **Storage in FAT table

✓ **+ 3 pts** *(linked) list of blocks, each block containing list of rows*

+ 2 pts Partial answer

+ 0 pts Unattempted/Incorrect

**#2- **Contents of row

✓ **+ 3 pts** *name, attributes, pointer to the first block/cluster etc.*

+ 2 pts Partially correct answer

+ 0 pts Unattempted/Incorrect

**#3- **New file creation

✓ **+ 4 pts** *Correct and detailed explanation of the process with data structures and design*

- 1.5 pts Haven't mentioned how new blocks are allocated

+ 1 pts Partially correct answer / incorrect justification

+ 0 pts Unattempted/Incorrect

QUESTION 6

Q6 15 pts

6.1 (a) **3 / 3**

+ 0 pts Incorrect/ not attempted

✓ **+ 3 pts** *To prevent the waiting for a per-queue lock, two different queues, per-cpu software queue (lockless) and hardware queue, are used. Requests for a cpu are first stored to its software queue and then the same requests are sent to the hardware queue which may have requests from other cpus too.*

+ 2 pts Incomplete answer

+ 2 pts Partially correct

+ 0 pts Incorrect answer (software and hardware queues are used for storing different types of requests).

+ 1 pts Incomplete answer

6.2 (b) **3 / 3**

✓ **+ 3 pts** *To have a large set of requests for optimization*

+ 2 pts Other reasons

+ 0 pts Not attempted/ incorrect

+ 2 pts Incomplete/partially correct

6.3 (c) **2 / 2**

✓ **+ 1 pts** *It corrupts the file system.*

✓ **+ 1 pts** *Because there may be some pending requests*

+ 0 pts Not attempted/ incorrect

+ 2 pts Correct

6.4 (d) **6 / 7**

+ 0 pts Incorrect/ not attempted

+ 2 pts Incomplete answer

+ 4 pts Create a duplicate copy of the USB or FAT file system (very inefficient ad add a lot of overheads in terms of storage)

✓ **+ 6 pts** *record changes to the file systems before they are committed for faster recovery in case of corruption*

+ 1 pts Enable write-through caching memory in file system

+ 3 pts partial correct

QUESTION 7

Q7 15 pts

7.1 (a) 5 / 5

✓ **+ 5 pts** *Correct*

+ 3 pts partially correct

+ 0 pts Incorrect/Not attempted

- 1 pts Minor Point missing/Error

7.2 (b) 5 / 5

✓ **+ 5 pts** *Correct*

+ 0 pts Incorrect/Not Attempted

- 1 pts Minor Error

+ 3 pts Partially Correct

7.3 (c) 3 / 5

+ 5 pts Correct

+ 0 pts Incorrect/Not Attempted

✓ **+ 3 pts** *Partially Correct*

- 1 pts Minor error

Entry Number: 2020CS10567 Name: VIRAJ AGASHE

COL 331/633

Major Exam
Semester II, 2022-23
Total marks: 80

Undertaking:

As a student of IIT Delhi, I will not give or receive aid in examinations. I will do my share and take an active part in seeing to it that others as well as myself uphold the spirit and letter of the Honour Code,

Signature: Viraj

Instructions:

1. All the questions are **compulsory**.
2. No doubts will be addressed during the exam. Make your assumptions.
3. Fill in your entry number and name on each page of the paper.

Questions:

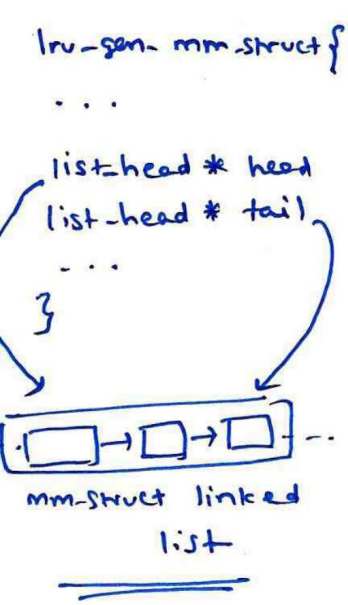
1. Answer the following questions: [3 + 3 + 4 Marks]
   a. What is the page walking process used for in the MG-LRU algorithm? Answer in the context of the *lru_gen_mm_state* structure.

In the MGLRU algo, the page walking process is used for the process of ageing pages.

When we perform the ageing process, we wish to find the pages whose access bit = 1 so that we can promote them to the (max-seq+1)th generation.

For this purpose, in the lru_gen_mm_struct, we keep a linked list (with 2 pointers) of mm_structs.

The mm_structs contain the base addresses of page tables (pgd's) of all processes in the NUMA nodes. We use this to walk the page table, check the pages' access bits & promote them if needed.



Entry Number: _____ Name: _____

b. How is a Bloom filter used to reduce the overhead of page walking?

During the process of evicting pages, in "lru-gen-look around", we check if (using the reverse map) the pages corresponding to the PMD entry of the evicted page (neighbouring pages) are all young (access bit=1). If yes, we promote them. Further, we add ~~them to the~~ the PMD address to the bloom filter if a lot of pages are young.

Then, during page walking for ageing, at the pmd level, we check if ~~PMD entry~~ that PMD address is in the bloom filter.

If NO, we can be sure it is mostly having old pages which we don't need to ~~age~~ boost up for ageing. So we can skip them. Thus, page walking becomes more efficient with bloom filter.

c. What is the need to deliberately mark actively used pages as "non-accessible"?

We need to track page accesses for the ageing & eviction algo to work. However, it is not possible for every memory access to be associated with book-keeping — performance hit! (If there is an interrupt with each page access etc.)

So, we periodically mark actively used pages and set ~~with~~ their ACCESS bit = 0. The next time this page is accessed:

→ Either a page fault occurs. The kernel can ~~track~~ note that it was accessed and set access bit = 1.

→ Or, the hardware provides some support and sets the access bit to 1.

Essentially, in the next round of ageing & eviction, the access bit of each page can be thus checked to see whether it was accessed recently or not, and then promote it to higher generation on the basis of whether it was.

Entry Number: 2020CS10567 Name: Viraj Agarwal

2. What is the *swappiness* variable used for, and how is it normally interpreted? When would you prefer evicting FILE pages as opposed to ANON pages, and vice versa? Explain the latter question with use cases. [4 + 6 Marks]

complete

The swappiness variable is a hyperparameter (typical value = 60) which is used in the "get_type_to_scan" function of the kernel in the MGLRU algorithm. The swappiness value helps us to determine which type of pages to evict from the min-seq generation — FILE or ANON pages.

→ In general, a high value of swappiness ⟺ we prefer to evict anon pages. Some specific values of swappiness — (Denote by swp for conciseness)

- swp = 0/1 ⇒ FILE
- ~~If minseq[FILE] < minseq[ANON]~~ swp = 200 ⇒ ANON
- other values of swp: call the "get type to scan" function.

- The swappiness parameter is used in the PI controller feedback loop. Essentially, the controller sees the refault rates for ANON & FILE pages and normalizes them by using this swappiness parameter. A high value of swappiness will lessen the contribution of ANON pages to the refault rate, & so the feedback loop is more likely to select ANON, even if the refault rate (absolute) is higher.

Use cases: maybe in some ~~programs~~ nodes (NUMA nodes) we are doing computation intensive work & node much file I/O. So, we would prefer file pages to be evicted rather than pages containing variables etc. which would slow down our computations.

Initially in isolate_folios which calls "get_type_to_scan" if can't break ties. (based on minseq & swappiness)

COL331/COL633 80 Marks

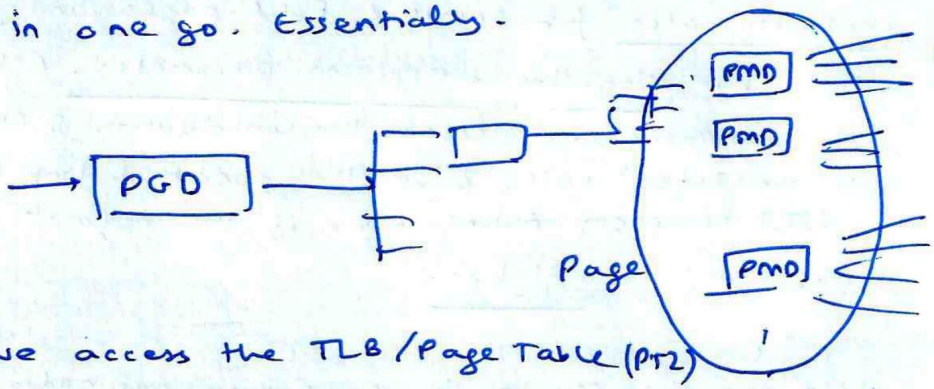
Entry Number: _____ Name: _____

3. Let us say that you want to "page" the page table. In general, the page table is stored in memory, and it is not removed or swapped out – it is basically pinned to memory at a pre-specified set of addresses. However, now let us assume that we are using a lot of storage space to store page tables, and we would like to page the page tables such that parts of them, that are not being used very frequently, can be swapped out. Use concepts from *folios*, *extents*, and *inodes* to create such a swappable page table. [10 Marks]

Note that page tables are inherently multi-level in nature in Linux. ~~This makes making them multi~~ This makes paging them easier. ~~We can keep~~

For example, it is possible that certain PMD entries are being accessed frequently while for others they are not.

Therefore, we can page the page table at some level. Let us say PMD level for now. Thus, we create a page table pinned to memory → called it PT2 which pages the PMDs (i.e. the 4th level of the page table). We can store the PMDs in a folio, so that we can load it contiguously in one go. Essentially.



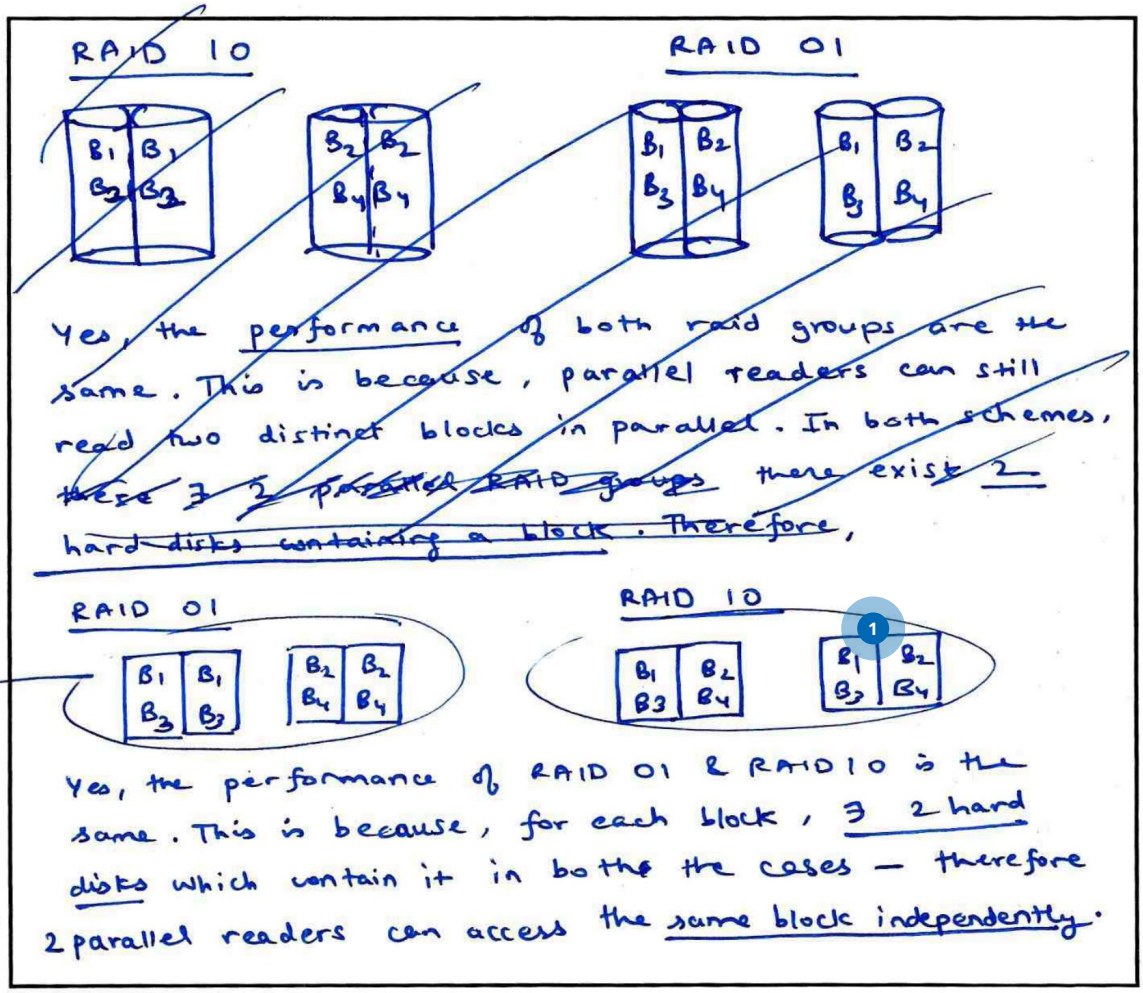
We access the TLB / Page Table (PT2) for the address of the PMD. If it is not in the main memory, ~~wait the PT2~~ bring it in from the swap space to the main memory & update mappings in PT2.

(We need not do this only for PMD, we can do at a higher level as well, depending on resource constraint)

Entry Number: 2020CS10567 Name: VIRAJ ASHISH

In this way, we use PT2 to page the page table of a process..

4. RAID 0 stripes data – odd numbered blocks in disk 0 and even numbered blocks in disk 1. RAID 1 creates a mirror image of the data (disk 0 and disk 1 have the same contents). Consider RAID 10 (first mirror and then stripe), and RAID 01 (first stripe and then mirror). Both the configurations will have four hard disks divided into groups of two. Each group is called a first-level RAID group. We are essentially making a second-level RAID group out of two first-level RAID groups. Now, answer the following questions:
   a. Is the performance (as we have defined it in class or as is commonly defined in terms of the bandwidth) of RAID 01 and 10 the same? [4 Marks]



Note that the hard disk locations (wrt groups) is diff, but the same no. of [1/3] [1/3] in both, etc. So, identical distribution if considered "across" groups.

COL331/COL633 80 Marks

Furthermore, though the distribution amongst groups is different, if we compare disks across groups, then the same amount & distribution of blocks exist (hard disks have same data & bandwidths collectively, only their distribution among groups is different, as seen from the previous figure). So, the bandwidth/performance remains the same.

b. What about their reliability? Is it the same? You need to make an implicit assumption here, which is that it is highly unlikely that both the disks belonging to the same first-level RAID group will fail simultaneously. [6 Marks]

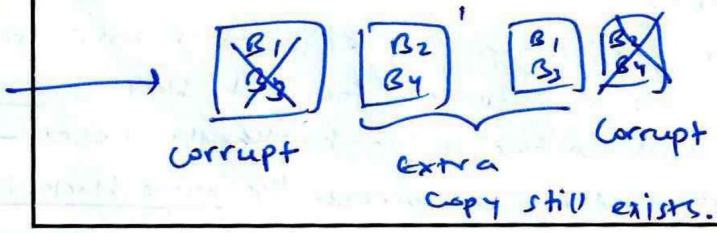
No, the reliability of RAID 10 & RAID 01 is not the same. Consider the same figure as for the previous part. Note that in RAID 10, the disks containing [B1 B3] belong to the same group while in RAID 01 these are on different groups.

If we assume that it is unlikely that disks of the same RAID group fail together, then we can conclude that RAID 10 is more reliable. This is because, since the first RAID group is mirror, 2 disks in a group have the same info, & since unlikely to fail together, we will always have a safe copy of data in the storage.

On the other hand, in RAID 01, since first group is stripe, disks which contain same blocks (mirror) are in different groups. It is possible due to some independent events, say hard disks containing [B1 B2] in both groups get corrupted. Then we have lost the data! Therefore, RAID 10 is more reliable than

We assume this to be more likely than same first level group to fail simult.

If this happens in RAID 10, we still have another copy of both corrupted disks!

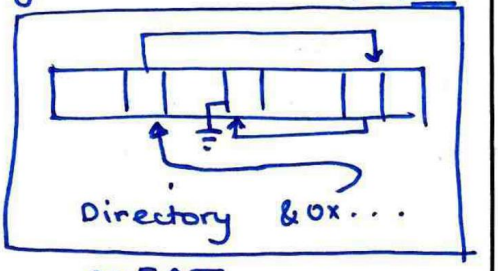


Entry Number: 2020CS10567 Name: VIRAJ AGASHE

Consider a large directory in the exFAT file system. Assume that its contents span several blocks. How is the directory (represented as a file) stored in the FAT table? What does each row in a directory's data block look like? How do we create a new file and allocate space to it in this filesystem? For the last part, explain the data structures that we need to maintain. Justify the design. [3 + 3 + 4 Marks]

The exFAT file system consists of a linked list embedded in array.

The FAT table would store a pointer to the base address of the directory (which is also a file).



exFAT

The base address in the array contains a block of data, and the pointer to the next block of data in the file. This continues for every block till the file ends.

In our case, the directory will essentially have a lot of subdirectories, or files. So, the data block of a directory would essentially look like a linked list of direntry structs.

These themselves would have pointers to the inodes as well as for the files (in the FAT fs itself) i.e. Another and other info

The direntries themselves would have pointers to the inodes, sibling & subdirectories, etc. as well as to files.

For allocating a new file, one possible idea is to keep a linked list of unmapped (empty) blocks in the FAT table array. Thus, finding the empty block to allocate is as easy as getting the first element of the LL and updating pointers (for head, etc).

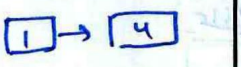
Entry Number: _____ Name: _____

So initially the LL. will have addresses of all unmapped blocks & gradually we keep selecting blocks from this. The advantage is that initially at least, files will be allocated contiguously (like the concept of folios) since the LL will have blocks in increasing order of addresses.

The Q is: when deleting a file what do we do?

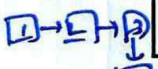
If we are fine with a slower filesystem, we can insert the freed blocks in sorted order in the LL (i.e. in their correct positions) which will make contiguous allocation easy. Otherwise if time is an issue then we can just insert at the end: O(1) but prevents contiguous allotment for large files

eg.

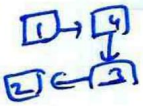


Suppose 2,3 are freed.

Opt 1:



Opt 2:



6. Answer the following questions with respect to devices and device drivers:

a. Why do we have both software and hardware request queues in *struct request_queue*? [3 Marks]

We have both software & HW request queues because the software queues behave as a "frontend" queue. First there are per-CPU queues in which a single CPU sends the queries (without races). These are collected (using locked queue) in a common queue also in software. Here, we also perform basic merging / delaying operations, etc. This collective queue interfaces with the ~~hardware queue~~ and the I/O schedulers after which it is finally sent to the hardware (device) after performing suitable delay & merging operations. It is not possible to do this in hardware (not cheaply, at least!)

b. Why do device drivers deliberately *delay* requests? [3 Marks]

Drivers deliberately delay I/O requests to the I/O device (block devices), since if we delay requests, it is possible that we may receive more requests for the same block, or nearby blocks.

Therefore, delaying requests gives us the chance to improve performance by merging adjacent queries / only have to read a block once, i.e. increases scope for optimization.

c. Why should we just not remove (eject) a USB key? [2 Marks]

We should not remove USB keys directly, because the device driver may have delayed I/O requests to it, which may still be pending. Further we often cache writes to the USB devices which we may need to write back. This could corrupt the data if not written back.

d. What can be done to ensure that even if a user forcefully removes a USB key, the FAT file system is not corrupted? [7 Marks]

The FAT file system could get corrupted (i.e. have incorrect data) if we remove a USB forcefully, due to cached writes which are pending. These may also include partial updation of the file system — we may have been in the process of adding a new entry, etc.

→ To resolve this, the device driver for that USB key can store this information & data (eg. that there are writes pending, what changes to be made to FAT fs, etc.) in as some metadata on the disk.

The next time the USB key is inserted, the device driver can check if everything is in order. If not, it fixes the file system by reusing the information which it had stored.

store pages which were not written back etc.

Entry Number: _____ Name: _____

7. Answer the following questions with respect to virtual machines: [5 + 5 + 5 Marks]

a. How does the host OS keep track of updates to the guest OS's page tables?

We know that there are 2 main ways in which the host OS can do this —

① Nested Paging: In this mode, first the guest VA GVA → GPA, and then the GPA → HPA by the host page table. In this paradigm, host OS is involved in the address translation, and so any new mapping etc. in the guest page table will raise a page fault, and the hypervisor can use this opportunity to create this mapping in its own page table (GPA → HPA). We don't need to change GPA → HPA mappings once allocated since phy addresses do not change.

② Shadow Paging: In shadow paging, the host keeps a shadow page table for the mapping from GVA → HPA. In this case the hypervisor can ~~simply map the page table to the~~ translate the GVAs directly. It can keep the shadow PT updated by:

→ Whenever a new page mapping is done in the guest, & it tries to access it, a page fault will occur, the hypervisor can in this case create the mapping in its own shadow table.

→ Further it can check the guest page table (which only serves the purpose of keeping shadow page table updated) — if any mappings have been updated / changed, we modify the same in the shadow page tables.

· Essentially on TLB miss / page fault, the VMM / Hypervisor can use the opportunity to update its mappings.

b. If there is a context switch in the guest OS, how can the host OS get an idea about the id (or something equivalent) of the new process (one that is being swapped in)? Even if the host OS is not able to find the *pid* of the new process being run by the guest OS, it should have some information available with it such that it can locate the page table and other bookkeeping information corresponding to the new process.

We know that some bits of the virtual address are ~~received for~~ associated with the PCID/ASID (process context ID) which essentially helps to identify which process/ thread group that address belongs to, for the TLB.

Since the hypervisor needs to keep a separate page table (shadow) for each different process in the guest OS, it needs to know when a context switch occurs, so that it can switch to the appropriate shadow PT for that process and not continually use the old one.

Since the hypervisor is essentially involved in the address translation (since the guest OS does not have privilege 0 in most cases), we can do this easily.

We know that CR3 holds the base address of the page table. So when there is a context switch, the guest OS will try to change CR3, which it has no permission to do.

When the VMM comes in, it will see that the value of CR3 was being changed. Further since bits 11:0 of the CR3 have the PCID, the VMM can read these. Further the VMM can create a map from PCID ⟶ Shadow PT base address.

Thus, on a context switch, from the value the guest OS process tried to write to CR3, we can get the PCID, load appropriate page table, and do the book-keeping. TLB entries need not be updated since GVAs will have the PCIDs.

(c) Why does HW-assisted virtualization allow the host's shadow page table and the guest page table to go out of sync, and why is this not a problem?

Since the hardware is directly involved in HW assisted virtualization, we can directly subvert the guest page tables and use the page table (shadow) of the VMM as soon as the base address of the PTE is modified in the CR3 register.

Therefore, any changes can be done directly to this shadow PT, and we don't need to keep it in sync with process PT.

This is only possible if the HW assists ~~virtual~~ virtualization, else we need an additional software layer between the guest OS and the shadow PT.