Lecture #11 CS695

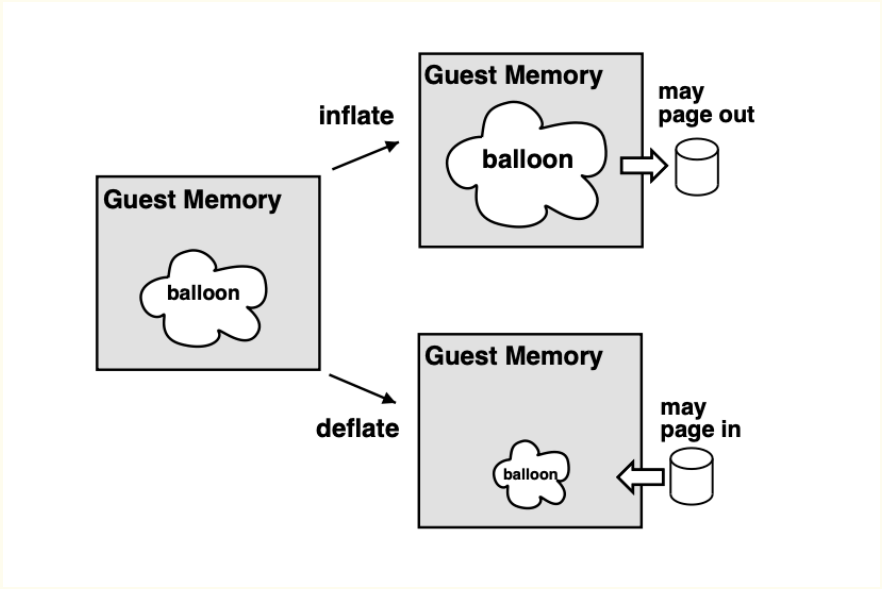
Memory Resource Management

- primitives
- mgmt. policies / decisions.
  - content-based

ballooning, sharing, migration, checkpointing, record & replay

(i) A2 due today

(ii) Quiz 1 on Friday 14th Feb. 9.30 am



- symbiotic: pressure by VMM, eviction by guest OS
- pinning
  - to avoid reclaimation
- all ballooned pages in guest valid=1, Present=0 } pte flag bits

Q. is this experiment reproducable?

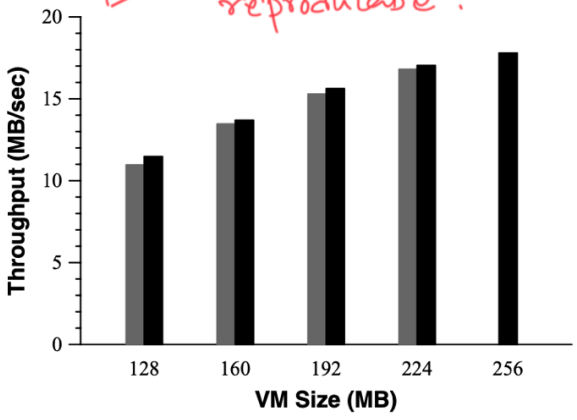


Figure 2: **Balloon Performance.** Throughput of single Linux VM running `dbench` with 40 clients. The black bars plot the performance when the VM is configured with main memory sizes ranging from 128 MB to 256 MB. The gray bars plot the performance of the same VM configured with 256 MB, ballooned down to the specified size.

(*) what was workload pattern? of each of the 40 clients.

or removed from a VM in order to rapidly adjust its

Sharing (content based page sharing)

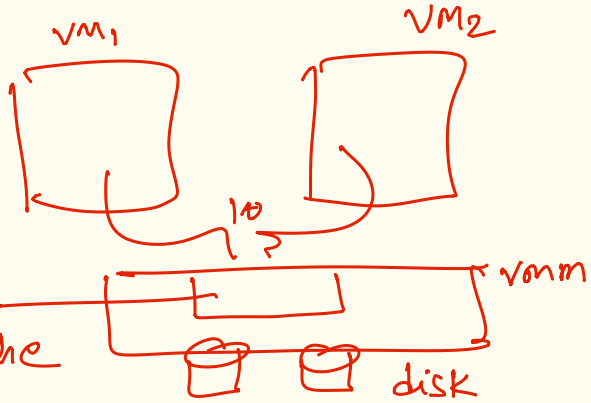
~ same memory regions / pages across VMs is a common possibility

soln: ~~dedup~~ ~~dedup~~ deduplicate!

~ processes: code, data, heap, stack.

~ VMM-view: ppa → mpa (no semantics of page use)

soln: ~ intercept disk IO calls (device virtualization)



block <23, 64, 716 846>

cached object *

---

memory sharing of VMs.

~ identify similar pages by content

~ hypervisor has access to all p2m mappings of all VMs

~ scan pages periodically / background...

~ for each scanned page
- hash the page
- lookup if hash exists
- if yes
  - ① recompute hash of hint page & compare
  - ② byte-by-byte compare of pages.
    - P_1 to m
    - P_2 to m
    - } share a m/c page
    - mark as CoW
    - ref count

else, add hash to a hint set

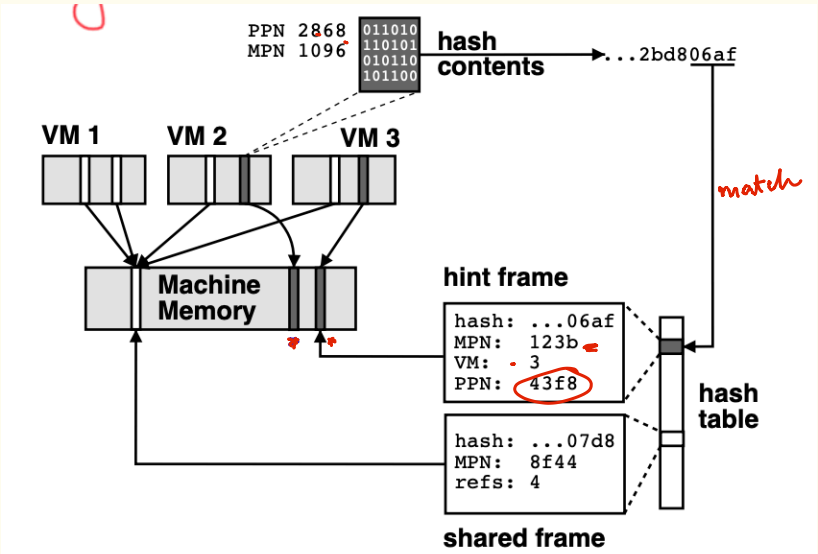


Figure 3: **Content-Based Page Sharing.** ESX Server scans for sharing opportunities, hashing the contents of candidate PPN 0x2868 in VM 2. The hash is used to index into a table containing other scanned pages, where a match is found with a hint frame associated with PPN 0x43f8 in VM 3. If a full comparison confirms the pages are identical, the PPN-to-MPN mapping for PPN 0x2868 in VM 2 is changed from MPN 0x1096 to MPN 0x123b, both PPNs are marked COW, and the redundant MPN is reclaimed.

Linux
- KSM — Kernel samepage merge
- madvise

questions CBPS operation
- when to scan?
- how much to scan?
- where to scan?

time to discover sharing potential ?

	Guest Types	Total MB	Shared MB	Shared %	Reclaimed MB	Reclaimed %
A	10 WinNT	2048	880	42.9	673	32.9
B	9 Linux	1846	539	29.2	345	18.7
C	5 Linux	1658	165	10.0	120	7.2

Figure 5: **Real-World Page Sharing.** Sharing metrics from production deployments of ESX Server. (a) Ten Windows NT VMs serving users at a Fortune 50 company, running a variety of database (Oracle, SQL Server), web (IIS, Websphere), development (Java, VB), and other applications. (b) Nine Linux VMs serving a large user community for a nonprofit organization, executing a mix of web (Apache), mail (Majordomo, Postfix, POP/IMAP, MailArmor), and other servers. (c) Five Linux VMs providing web proxy (Squid), mail (Postfix, RAV), and remote access (ssh) services to VMware employees.

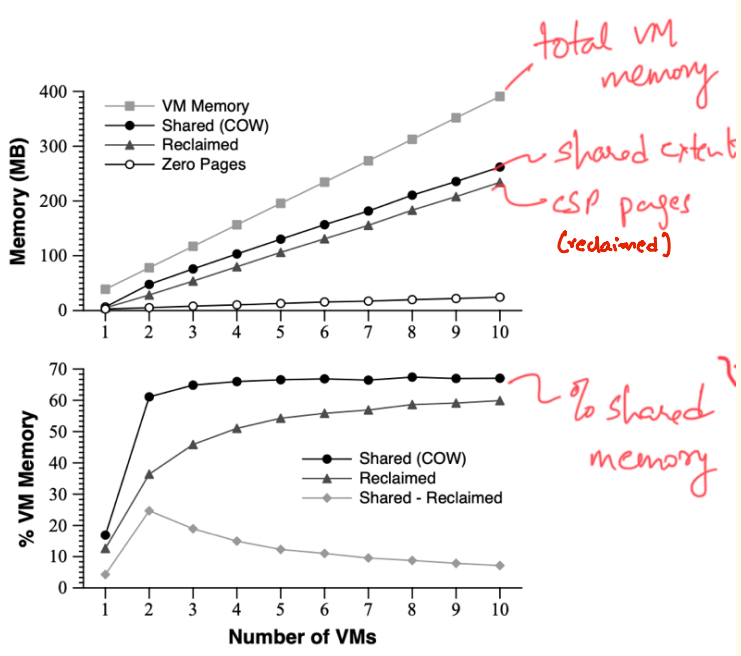
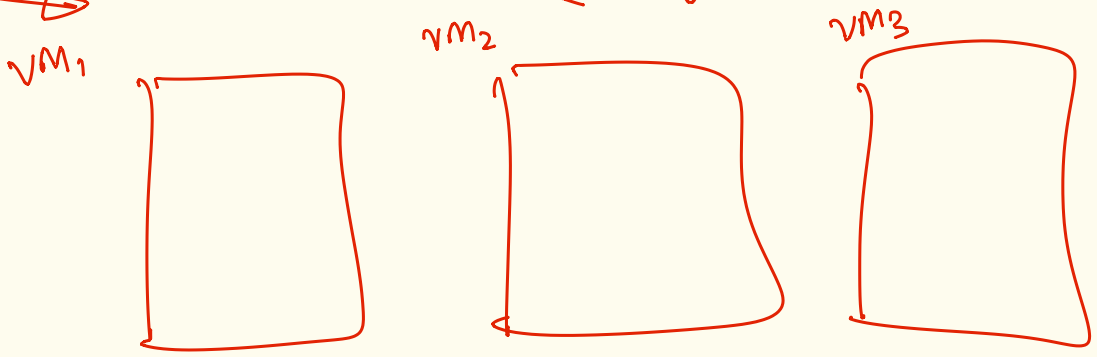


Figure 4: **Page Sharing Performance.** Sharing metrics for a series of experiments consisting of identical Linux VMs running SPEC95 benchmarks. The top graph indicates the absolute amounts of memory shared and saved increase smoothly with the number of concurrent VMs. The bottom graph plots these metrics as a percentage of aggregate VM memory. For large numbers of VMs, sharing approaches 67% and nearly 60% of all VM memory is reclaimed.

order of usage of primitives for memory mgmt:

- (i) sharing
- (ii) ballooning
- (iii) demand paging / swapping.

how to decide memory allocation extents?



(#) balance performance & memory efficiency / utility.

scheme: WSS — working set size

↳ active memory region / area

(#) share-based allocation (proportionate)

$s_1 : 0.3$

$s_2 : 0.2$

$s_3 : 0.1$

effective priority / fraction for allocation → $$\phi = \frac{S}{P \cdot (f + K(1-f))}$$

S = #shares / currency of a VM

P = #pages

f = fraction of active pages

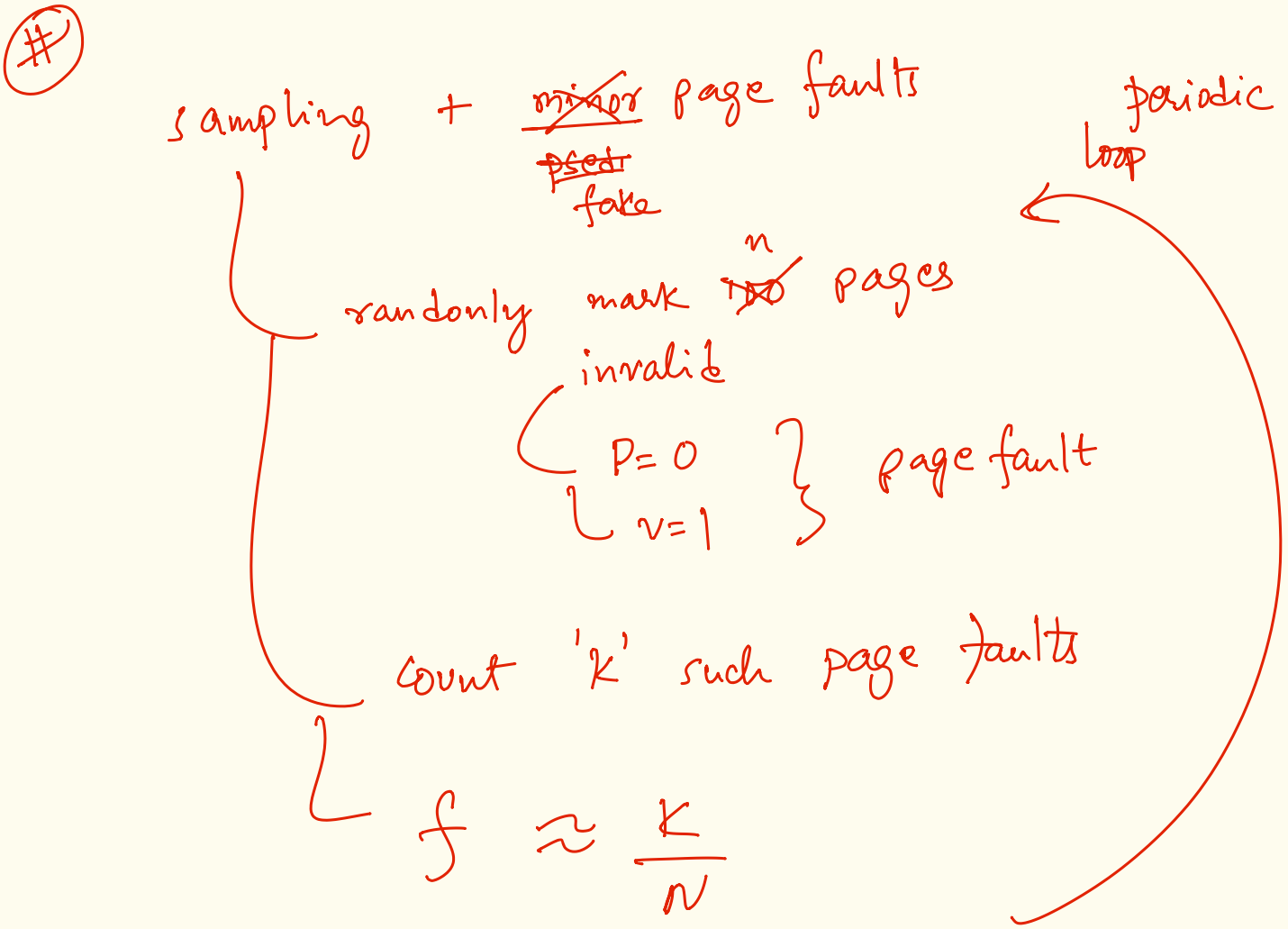
K = inactive tax. $K = \frac{1}{1-\tau}$

$0 < \tau \leq 1$

τ = tax rate $\phi = S/P$

$\tau = 0 \Rightarrow K = 1$

$= 1 \Rightarrow K = \infty \ \phi = 0$


#
sampling + ~~minor~~ ~~pseudo~~ fake page faults
periodic loop
randomly mark ~~1000~~ n pages invalid
P=0
V=1
page fault
count 'k' such page faults
$f \approx \frac{k}{n}$