Memory Ballooning

In case of a cloud offering, the resources are best utilized by overcommitment. The case for overcommitment is that not every user/VM will be using all of the resources allocated to it at the same time. So, by allocating more resources to the VMs than are available, cloud can support more number of VMs.

The Problem with simple memory overcommitment

When a Virtual Machine starts, it is allocated physical memory page frames from the host machine by the hypervisor on demand, similar to the processes running on an OS. But unlike processes, once a memory page is no longer needed by the guest OS, it does not return it back to the hypervisor. So, once memory is given to a guest OS, it cannot be reclaimed by the hypervisor. Why does this happen?

When an Operating System is running in the virtualized environment, it must get the illusion that all the memory starting from physical address 0 belongs to it. Multiple Operating Systems should be able to run on the same physical memory. To do this, hypervisor must add an extra layer of virtualization to virtualize physical memory for each VM on top of the machine memory. Hypervisor maintains a mapping from the physical pages of a virtual machine to the actual machine memory pages, similar to the page tables of a process. The guest OS internally keeps per process page tables. So, it takes two translations for the virtual address of a process running on guest OS to be translated to actual physical address of the machine memory.

When a Virtual Machine starts, none of its physical memory pages are mapped to the actual machine pages. These are allocated on demand as and when the hypervisor intercepts all the page faults and TLB misses of the guest OS.