Review : Efficient Live Migration of Virtual Machines Using Shared Storage

Published : VEE 2013.

***Authors* :**

Changyeon Jo, Erik Gustafsson, Jeongseok Son, Bernhard Egger .

***Goal* :**

To make live migration of VMs more efficient by reducing downtime, total migration time and reducing the total data transferred during migration.

***Problem Statement* :**

In live migration of VMs, it is important that both downtime and total migration time are less. Reduced downtime ensures that there is no disruption of active services and reduced total migration time ensures that the original host can be used for other purposes in less time. This make live migration efficient. But, till now, there is much focus on reduction in downtime at the expense of total migration time. So, can the downtime and the total migration time be simultaneously reduced ? How do we address this tradeoff ?

***Description* :**

- Limit on maximum bandwidth utilized for sending data between 2 physical hosts is needed to ensure quality of service. But this can increase the total migration time when more memory needs to be transferred. The Pre-copy approach to migration, though very useful for reducing downtime has shortcomings in terms of total data transferred and the total migration time. These problems can be addressed by reducing the amount of data transferred between the 2 hosts.
- The paper tries to reduce the total migration time by reducing the total data transferred between the two physical hosts. There is a lot of duplication of pages between memory and the disk. This duplication is the motivation to solve the problem. So, instead of sending the actual page in memory during migration, we send the memory to disk mapping so that the other host can fetch these pages directly from the Network Attached Storage using these mappings. This happens simultaneously along with the migration and hence reduces the total migration time.
- Along with the memory to disk mappings of duplicated pages, we also transfer unique memory pages directly from the source to the target.
- All I/O operations to the permanent storage need to be tracked to maintain an updated list of memory pages that are currently duplicated on the storage device.
- The above solution is applied to the pre-copy approach to migration. Pre-copy approach to migration is an iterative process where dirty pages from the previous iteration are transferred in this iteration. Now for duplicated dirty pages, we just send the memory to disk mappings. While for the rest of the dirty pages that are not duplicated, they are directly sent. NAS fetch queue at the target host takes care of these mappings and fetches the pages on disk to the migrated VM address space. The target host waits until the NAS fetch queue has processed all outstanding entries and then restarts the VM.
- Each VM has a p2b map. To maintain a consistent p2b map, the memory writes to memory pages in the p2b map are tracked and the page is removed from the map because after the write operation, there is no copy of the page on the disk.

***Positive Points* :**

- The paper focusses on total migration time which has been neglected by many techniques in their effort to reduce downtime.
- The reduction in data transferred between the 2 hosts can further be reduced by using compression techniques and thereby lessen total migration time further.
- The transparent I/O interception by the VMM makes sure that the contents of the page cache of VM are visible to the VMM so that the VMM can know which pages are duplicated in memory for creating the memory to disk mappings.

***Negative Points* :**

- One write access to the memory page in the p2b map removes the corresponding mapping from the map. This causes an overhead because the entire page has to be sent directly now because there is no duplication.
- Large amount of duplication can suffer from an increased downtime.
- This technique cannot be applied to live migration between hosts that do not share storage.
- The authors have done their experimentation on Xen hypervisor where they have assumed that the NAS is mounted in Domain 0 and accessed as a file-backed disk in the VM. I am not sure whether this is reproducible for other hypervisors.

***Future Scope* :**

- The target host waits until the NAS fetch queue has processed all outstanding entries and then restarts the VM. The VM can be made to restart sooner by discarding the NAS fetch queue if the queue is long and fetching as and when the requirement of the page not in memory arises.
- Another approach is to make the NAS fetch queue run in the background and the VM is restarted without waiting for all the pages in the queue to be fetched.