Efficient Disk Space Management for Virtual Machines

Abhishek Gupta (student), Norman C. Hutchinson
Department of Computer Science
University of British Columbia
{*agupta, norm*}*@cs.ubc.ca*

Abstract

There has been tremendous interest in virtualization infrastructure lately. Virtual Machines are finding use in scenarios ranging from kernel development to economic honeyfarm deployment. As we go about evolving the technology to fit our current application needs, there remain a variety of challenges to be overcome. One key challenge lies in efficient utilization of secondary storage systems.

From a file-system perspective, several interesting problems are exposed upon the instantiation of a new Virtual Machine (VM). Instantiating a new VM entails guaranteeing the availability of at least one fixed size disk partition. Further, several of these VMs, when instantiated first, require identical software images on their allocated disk space. Currently these problems are addressed by using block level storage abstractions which provide copy-on-write mechanisms for sharing of data. LVM, the Linux volume manager and Parallax are well acknowledged as current state-of-the art in this space. Each of these artifacts utilize variants of copy-on-write schemes for providing block level sharing between logical volumes and have their own peculiar shortcomings.

Our reservation with LVM is mainly with its aggressive copy-on-write implementation. Before a disk block is replaced from an original volume, LVM writes it to all existing snapshots. Naturally, providing support for these writes would overtly tax systems running scores of virtual machines relying on LVM snapshots as their back-end devices. As for Parallax, our primary concern is that its design mandates an expensive traversal through the entire height of a radix tree for single block access. Buffer-cache mechanisms might alleviate this problem to some extent but supporting benchmarks are still to be established. Another subtle problem with these designs is that although they provide excellent sharing of blocks between VMs, they fail to maintain the proximity of disk blocks on the physical disk. The benefits of this tradeoff between sharing and ease of classic read-ahead accesses warrants further investigation.

These problems are further escalated with the availability of fork in popular VM technology. The Potemkin team has well articulated the copy-on-write semantics for forking virtual machines; however, the use of a ram disk instead of an actual device is not a durable solution. To this extent LVM provides limited support as it does not allow the creation of recursive snapshots, whereas, Parallax provides an appealing solution, though at the cost of increased disk fragmentation, which further complicates block reclamation and linear accessibility.

It should be noted that naive block level sharing is not a complete solution in the context of forking VMs. It is likely that child VMs will have restricted access to the resources of their parent VMs, thus, necessiating the implementation of mechanisms to provide fine-grained protection over shared objects. To this extent, copy-on-write block level sharing is an obvious protection against malicious writes to the file-system, however, it fails to prevent malicious reads.

As of now we are evaluating the applicability of LVM and Parallax in the context of VMs. We have implemented a Parallax-like radix tree target in LVM and are benchmarking it along with the standard LVM linear target to empirically establish their limitations. Orthogonally, we are assessing if block level sharing is actually the correct solution for VMs. We feel that there is a need to re-evaluate the current state of the art in block level sharing and evolve it to support fine grained snapshots while maintaining efficiency in data access. Through our work-in-progress presentation we would like to share some experiences from our implementation and discuss other potential solutions.