

Violin: A Framework for Extensible Block-level Storage

Michail D. Flouris[†]

Department of Computer Science,
University of Toronto,
Toronto, Ontario M5S 3G4, Canada
flouris@cs.toronto.edu

Angelos Bilas[‡]

Institute of Computer Science (ICS)
Foundation for Research and Technology - Hellas
P.O.Box 1385, Heraklion, GR 71110, Greece
bilas@ics.forth.gr

Abstract

In this work we propose Violin, a virtualization framework that allows easy extensions of block-level storage stacks. Violin allows (i) developers to provide new virtualization functions and (ii) storage administrators to combine these functions in storage hierarchies with rich semantics. Violin makes it easy to develop such new functions by providing support for (i) hierarchy awareness and arbitrary mapping of blocks between virtual devices, (ii) explicit control over both the request and completion path of I/O requests, and (iii) persistent metadata management.

To demonstrate the effectiveness of our approach we evaluate Violin in three ways: (i) We loosely compare the complexity of providing new virtual modules in Violin with the traditional approach of writing monolithic drivers. In many cases, adding new modules is a matter of recompiling existing user-level code that provides the required functionality. (ii) We show how simple modules in Violin can be combined in more complex hierarchies. We demonstrate hierarchies with advanced virtualization semantics that are difficult to implement with monolithic drivers. (iii) We use various benchmarks to examine the overheads introduced by Violin in the common I/O path. We find that Violin modules perform within 10% of the corresponding monolithic Linux drivers.

1. Introduction

Storage is becoming an increasingly important issue as more and more data need to be stored either for archival or online processing purposes. As the amount of storage required increases, scalable storage systems provide a means of consolidating all storage in a single system and increasing storage efficiency. However, storage consolidation leads to increased requirements for “flexibility” that will be able to serve multiple applications and their diverse needs. This flexibility refers to both storage management and application access issues and is usually provided through virtualization techniques: Administrators and applications see various

types of virtual volumes that are mapped to physical devices but offer higher-level semantics through virtualization mechanisms.

The quality of virtualization mechanisms provided by a storage system affects storage management complexity and storage efficiency, both of which are important problems of modern storage systems. Storage virtualization may occur either at the filesystem level or at the block level. Although both approaches are currently being used, our work addresses virtualization issues at the block-level. Storage virtualization at the filesystem level has mainly appeared in the form of extensible filesystems [11, 24, 30, 32].

We argue that the importance of block-level virtualization is increasing for two reasons. First, certain virtualization functions, such as compression or encryption, may be simpler and more efficient to provide on unstructured fixed data blocks rather than variable-size files. Second, block-level storage systems are evolving from simple disks and fixed controllers to powerful storage nodes [1, 9, 10] that offer block-level storage to multiple applications over a storage area network [23, 22]. In such systems, block-level storage extensions can exploit the processing capacity of the storage nodes, where filesystems (running on the servers) cannot. For these reasons and over time, with the evolution of storage technology a number of virtualization features, e.g. volume management functions, RAID, snapshots, moved from higher system layers to the block level.

Today’s block-level storage systems provide flexibility in managing and accessing storage through I/O drivers (modules) in the I/O stack or through the filesystem. For instance, Linux-based storage systems use drivers, such as MD [2] and LVM [25] to support RAID and logical volume management functions. However, this flexibility is limited by the fact that current I/O stacks require the use of monolithic I/O drivers that are both complex to develop and hard to combine. As a result, current block-level systems offer predefined virtualization semantics, such as virtual volumes mapped to an aggregation of disks or RAID levels. In this category belong both research prototypes [2, 6, 8, 18, 25] and commercial products [4, 5, 12, 27, 28]. In all these cases the storage administrator can switch on or off various features at the volume level, but cannot extend them.

In this work we address this problem by providing a kernel-level framework for (i) building and (ii) combining virtualization functions. We propose, implement, and evaluate *Violin* (Virtual I/O Layer INtegration), a virtual I/O framework for commodity storage nodes that replaces the current block-level I/O stack with an improved I/O hierarchy that allows for (i) easy extension of the

[†]Work partly performed while at the Institute of Computer Science (ICS), Foundation of Research and Technology – Hellas (FORTH), P.O. Box 1385, Heraklion, GR 71110, Greece.

[‡]Also, with the Department of Computer Science, University of Crete, P.O. Box 2208, Heraklion, GR 71409, Greece.

