Making the most of your SSD: a case for Differentiated Storage Services

Michael Mesnier, Scott Hahn
Intel Corporation
{michael.mesnier, scott.hahn}@intel.com

Brian McKean et al.
LSI Corporation
{brian.mckean}@lsi.com

Abstract

Differentiated Storage Services (DSS) is a new QoS architecture for file and storage systems. DSS defines an OS interface by which file systems can assign arbitrary policies (performance and/or reliability) to I/O streams, and it provides mechanisms that storage systems can use to enforce these policies. The approach assumes that a stream identifier can be included in-band with each I/O request (e.g., using the Group Number field in the SCSI command set) and that the policy for each stream can be specified out-of-band through the management interface of the storage system.

This work-in-progress talk will present a case for the Differentiated Storage Services architecture. In particular, we will describe an early DSS prototype, based on Ext3 and a hybrid storage system composed of rotating and solid-state disks. With very little modification, Ext3 can identify latency-sensitive I/O streams (small files, directories, metadata, and the journal) and request that the storage system provision the solid-state storage for just these streams; and this is just one of many possibilities. As part of our ongoing work, across a variety of file and storage systems, we are looking into other QoS polices/mechanisms that can be used to improve application performance or reliability.

Extended abstract

Storage systems export a narrow I/O interface (ATA or SCSI) whose access to data consists primarily of two commands: READ and WRITE. This block-based interface abstracts storage from higher-level constructs, such as applications, processes, threads, and files. Although this allows operating systems and storage systems to evolve independently, achieving end-to-end application QoS can be a difficult task. In practice, applications with performance or reliability constraints will be assigned to dedicated, specially configured storage systems. However, storage consolidation techniques are often employed to reduce costs, and multiple applications are assigned to the same storage system. Such a one-size-fits-all approach requires a general purpose configuration, so application performance may not be optimal.

Storage consolidation is not a new problem, but the increasing heterogeneity of storage (e.g., low-cost vs. enterprise, rotating vs. solid-state) and the manner in which it is configured (e.g, nearline vs. offline, RAID level) is making it a bigger issue. Consider the task of integrating a solid-state disk into a disk array. One approach would be to use the disk to cache the most performance-critical blocks — but which blocks are these? With no awareness of applications and their I/O requirements, particularly in a consolidated environment where the I/O from numerous applications is merged, intelligent provisioning at the block level can be challenging.

One solution could be to tightly couple the QoS policies of an OS with the internal mechanisms of a storage system (e.g., caching), but this is not the best solution if the OS is to be interoperable across different storage systems. A better approach would be to separate policy from mechanism, where an OS is able to associate a desired QoS with each I/O stream — one that a storage system is able to interpret and support. To continue with the previous example, a solid-state disk could be used to cache the I/O of high-priority streams, but it is left to the OS to define each stream and determine its priority.