Enhancing the Linux Memory Architecture to Support File Systems over Heterogeneous Devices

Alexandros Batsakis, Randal Burns
The Johns Hopkins University

Thomas Talpey, Arkady Kanevsky, James Lentini
Network Appliance Inc.

The Linux kernel must deal with the ever-growing performance heterogeneity of network and I/O devices. In a heterogeneous environment a single, policy-based framework for memory management does not provide good write performance to all storage resources. Currently, Linux treats all memory pages uniformly without considering the capabilities of the underlying device.

New implementations of traditional file sharing mechanisms such as zero-copy NFS over RDMA make this problem more apparent. We have conducted a series of experiments using NFS over RDMA that show that write throughput for cached I/O lags far behind the available bandwidth. This is an indication that the current memory management scheme is not optimized for low-latency, high-bandwidth interconnects such as Infiniband or 10Gb Ethernet. Altough the memory manager can be bypassed via direct I/O, this solution requires changes to existing applications and often yields lower system performance because it loses the benefit caching.

We have identified the current flushing mechanism as the main reason for the write performance mismatch. Linux 2.6 invokes *pdflush* when the ratio of the dirty over the clean pages in RAM exceeds a predefined threshold that aggressively writes the dirty data to the backing storage device. This produces bursty write-backs, which forces the server to process a large amount of data in small time intervals. As a result, the server's CPU and bandwidth exhibit long periods of inactivity followed by periods of overloading. This overloading interferes with the server's ability to service I/O requests promptly on behalf of its clients. Although *pdflush* parameters can be tweaked to allow more fine-grained operation, they do not provide enough flexibility. This is because they are system-wide and they do not take into account the individual characteristics of each system (memory capacity) and device (bandwidth and latency).

In addition, many file system clients, such as NFS, attempt to augment the virtual memory operation by implementing their own mechanism for detecting memory pressure and flushing data. However, this behavior is still not suitable in all circumstances. Our measurements show that although the NFS policy works reasonably for Fast-Ethernet, it is less efficient for a 10Gbps Infiniband network. Also, this behavior interferes with the memory manager's ability to coordinate the different memory writers and prevent the exhaustion of memory pages that results in slow, synchronous writes.

We propose a series of enhancements to the Linux I/O model that include the memory system in the optimization path. These include a Linux page management system that is (1) *storage aware* in that it writes out memory pages differently depending upon the properties of the target device or server, (2) *system aware* in that it considers the memory pressure, memory capacity, and processor utilization in order to minimally interfere with other processes, and (3) *adaptive* in that it recognizes and adjusts to temporal variations in the performance of devices, networks, and servers. We are also investigating possible modifications to the NFS file system client and its interaction with the memory manager. Finally, the proposed paths are not bound to a specific storage protocol; they apply to any file system with an emphasis on network storage systems that transfer data over high-speed networks.