Abstract

We live in age of data ubiquity. Even the most conservative estimates predict exponential growth in produced, transmitted and stored data. Big data is used to power business analytics as well as to foster scientific discoveries. In many cases, explosion of produced data exceeds capabilities of digital storage systems. Scientific high-performance computing environments cope with this problem by utilizing large, distributed, storage systems. These complex systems can only provide a high degree of reliability and durability by means of data redundancy. The most straight-forward way of doing that is by replicating the data over different physical devices. However, more elaborate approaches, such as erasure coding, can provide similar data protection while utilizing less storage. Recently, software-defined reliability methods began to replace traditional, hardware-based, solutions. Complicated failure modes of storage system components also warrant checksums to guaranty long-term data integrity. To cope with ever increasing data volumes, flexible and efficient software implementation of error correction codes is of great importance. This thesis introduces a method for realizing a flexible Reed-Solomon erasure code using the "Just-In-Time" compilation technique. By exploiting intrinsic arithmetic redundancy in the algorithm, and by relying on modern optimizing compilers, we obtain a throughput-efficient erasure code implementation. Additionally, exploitation of data parallelism is achieved effortlessly by instructing the compiler to produce SIMD code for desired execution platform. We show results of codes implemented using SSE and AVX2 SIMD instruction sets for x86, and NEON instruction set for ARM platforms. Next, we introduce a framework for efficient vectorized RAID-Z redundancy operations of ZFS file system. Traditional, table-based Galois field multiplication algorithms are replaced with custom SSE and AVX2 parallel methods, providing significantly faster and more efficient parity operations. The implementation of this framework was made publicly available as a part of *ZFS on Linux* project, since version 0.7. Finally, we propose a new erasure scheme for use with existing, high performance, parallel filesystems. Described reliability middleware (ECCFS) allows definition of flexible, file-based, reliability policies, adapting to customized user needs. By utilizing the block erasure code, the ECCFS achieves optimal storage, computation, and network resource utilization, while providing a high level of reliability. The distributed nature of the middleware allows greater scalability and more efficient utilization of storage and network resources, in order to improve availability of the system.