Towards A Scalable, Resilient, and Efficient Data Service for Exascale Computing

Michael Brim[1], Tonglin Li[1], Sarp Oral[2], Feiyi Wang[2], Geoffroy Vallee[2], Scott Atchley[2]
[1]Computer Science and Mathematics Division, [2]National Center for Computational Sciences
Oak Ridge National Laboratory, Oak Ridge, USA

I. Introduction

Exascale high performance computing (HPC) systems are expected to include multi-tier storage systems. Burst buffers, consisting of fast non-volatile memory technology, will be a critical tier for resilient, high performance storage. A majority of the discussion surrounding burst buffers relates to staging of scientific simulation checkpoints, primarily the buffering of written data to allow applications to continue computation while the checkpoint is drained to a parallel file system. Such a narrow focus ignores read-intensive workloads (e.g., Big Data analytics and runtime visualization) that are expected to constitute an increasingly larger fraction of HPC jobs. The Exascale Computing Data Service (ExCDS) project is investigating a distributed data service that efficiently manages data transfers across application memory, burst buffers, and parallel file systems for a diverse collection of simultaneous workloads. In providing a center-wide data service, our goals are: (1) to relieve applications of the burden of explicit data management across storage tiers, and (2) to manage aggregate I/O behavior to improve utilization of storage resources. Our hypothesis is that by monitoring the storage hierarchy across all tiers, the data service can identify resource contention and I/O load imbalance and use that information to improve system-wide storage behavior through a collection of intelligent I/O management techniques. In comparison to existing techniques that optimize per-workload storage using I/O middleware, our solution has the potential to provide significant boosts to performance and productivity for all workloads sharing a multi-tier storage system.

II. Solution Architecture and Progress

There are four crucial components in our exascale data service architecture.

Application interfaces: We have investigated appropriate application-level interfaces and semantics for a scalable and resilient data service that supports both traditional scientific and Big Data workloads. The ExCDS application API uses three core data abstractions: (1) a data object, (2) a data object collection, and (3) a data namespace. These abstractions form an organizational hierarchy wherein a namespace is a container for data object collections and collections hold data objects. A data object is a contiguous region of bytes associated with a unique identifier within the enclosing collection. Operations on data objects are asynchronous. Collections are versioned to support time-evolving data and consistent data views from distributed application processes. Once committed, the data within a specific version of the collection is immutable. Collection attributes may be specified that influence data management decisions by the data service. Example attributes include hints such as expected I/O patterns, requirements for placement, persistence, or resilience, and desired quality of service (QoS). For compatibility with existing applications, we have also developed a POSIX I/O interposition layer that builds upon the ExCDS API. In future work, we plan to investigate the potential benefits of direct integration of our API within HDF5 and ADIOS.

System-wide Storage Monitoring: In prior work we have developed MELT [1], an advanced monitoring infrastructure for center-wide Lustre deployments. MELT uses a scalable overlay network called SNOflake for communication and distributed data processing. MELT collects host-based and Lustre metrics on clients, servers, and I/O routers, and then aggregates the metric data to provide system-level and job-level information. For this work, we have extended MELT to monitor burst buffer servers.

Versatile Data and Metadata Storage: ExCDS is designed to use a modular data management interface to backend storage technologies. The interface provides methods for storing and retrieving data and metadata associated with the core ExCDS data abstractions. Currently, we have implemented a POSIX file system data storage module and are developing modules for memory-based and burst buffer storage. ExCDS uses a key-value store (KVS) abstraction to manage metadata and runtime information (e.g., monitoring state). The current KVS implementation uses ZHT [2].

Intelligent I/O Management: The SmartIO component of ExCDS is tasked with using the information gained via system-wide monitoring to guide decisions on data placement, movement, and aggregation in the handling of distributed application API requests. This component also uses a modular design, so that alternative I/O management strategies can be implemented and compared. The current implementation use the DumbIO module that provides no advanced decision making and serves as a baseline. We are actively researching intelligent techniques based on analytical and machine-learning based I/O models to extend and improve upon our earlier work that studied balanced data placement and dynamic I/O resource path selection [3]. In future work, we plan to also study placement and movement decisions based on temporal access patterns, replication in support of data sharing, and system-level QoS mechanisms.

References

[1] Michael J. Brim and Joshua K. Lothian, *Monitoring Extreme-scale Lustre Toolkit*, International Workshop on The Lustre Ecosystem: Challenges and Opportunities, March 2015.

[2] Tonglin Li, Xiaobing Zhou, Ke Wang, Dongfang Zhao, Iman Sadooghi, Zhao Zhang, Ioan Raicu, *A Convergence of Key-Value Storage Systems from Clouds to Supercomputers*, Journal of CCPE, 2015

[3] Feiyi Wang, Sarp Oral, Saurabh Gupta, Devesh Tiwari, and Sudharshan Vazhkudai, *Improving Large-scale Storage System Performance via Topology-aware and Balanced Data Placement*, IEEE ICPADS 2014.