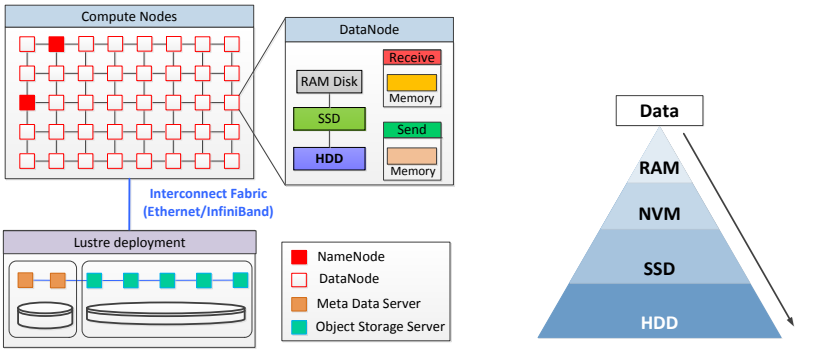
High Performance File System and I/O Middleware Design for Big Data on HPC Clusters

Nusrat Sharmin Islam, The Ohio State University, islamn@cse.ohio-state.edu
Advisor: Dhabaleswar K. (DK) Panda, The Ohio State University, panda@cse.ohio-state.edu

I. Problem Statement

Managing the proliferation of digital data in this Big Data era places a premium on high-throughput, high-availability storage. The most popular Big Data file system of recent times is the Hadoop Distributed File System (HDFS) which is well-known for its scalability and reliability. HDFS is the primary storage for a range of Big Data processing engines like Hadoop MapReduce, HBase, Hive, Spark, etc. HDFS, along with the Big Data Analytics frameworks and middleware, is increasingly being used on HPC clusters for scientific applications. Consequently, the wide adoption of HDFS makes its performance and scalability of critical importance to both Big Data and HPC community.



(a) HDFS Deployment (b) NVM in the Storage Hierarchy

Fig. 1: HDFS and Storage Hierarchy in HPC Systems

Modern HPC clusters are equipped with high performance interconnects like InfiniBand [5] that provide low latency and high throughput data transmission. The multi-core compute nodes have large memory along with heterogeneous storage devices like RAM Disk, SSD, and HDD [2]. Moreover, the traditional *Beowulf* architecture [13] model has been followed for building these clusters, where the compute nodes usually have limited-capacity local storage, while a sub-cluster of dedicated I/O nodes with parallel file systems, such as Lustre, is provided for data storage and access. Figure 1(a) shows an example of how heterogeneous storage devices and Lustre are deployed on most modern HPC clusters and how HDFS runs in this setup with the DataNodes being located on the compute nodes.

TABLE I: Performance vs capacity comparison for [2]

Type	Peak Bandwidth	Capacity
RAM Disk (local)	≈ 6.61 GBps	≈ 32 GB
SSD (local)	≈ 2.32 GBps	≈ 300 GB
HDD (local)	≈ 267.2 MBps	≈ 80 GB
Lustre	≈ 817.1 MBps	≈ 1.6 PB

All such heterogeneous storage devices on modern HPC clusters have different performance and storage characteristics. Table I shows the different types of storage media available in one of the leadership-class HPC clusters SDSC Gordon [2]. It is evident from this table that the amount of local storage spaces of RAM Disk, SSD, and HDD are negligible compared to the vast installation of Lustre. But in terms of data access peak bandwidth, RAM Disk and SSD demonstrate much better performances than Lustre and HDD. Moreover, emerging Non-Volatile Memory (NVM) system is paving its way into the HPC clusters. NVMs offer byte-addressability with near-DRAM performance and low power consumption for I/O-intensive applications. NVMs can not only augment the overall memory capacity, but also provide persistence while bringing in significant performance improvement. As depicted in Figure 1(b), NVMs are, thus, excellent contenders to co-exist with RAM and SSDs in large scale clusters and server deployments. This further implies that it is the software overheads that cause performance bottlenecks when using NVM to replace the disk-based storage systems.

The outstanding performance requirement in HPC environments pertains to unprecedented demands on the performance of supporting storage systems. The default design of HDFS cannot efficiently utilize the advanced features of the available resources in HPC platforms. As a result, HDFS suffers from poor communication performance as well as huge I/O bottlenecks due to the tri-replicated data blocks. Even though replication enhances data-locality, the requirement of large amount of local storage space due to replication makes the deployment of HDFS challenging on HPC platforms. It is, therefore, critical to re-think the architecture of HDFS for HPC systems. In this thesis, we address the following important challenges:

1) *Can we re-design HDFS to take advantage of high performance interconnects and exploit advanced features such as RDMA (Remote Direct Memory Access)? What are the challenges here?*

2) *How can we maximize overlapping among different stages of HDFS Write operation? Can we adopt the high-throughput Staged Event-Driven Architecture (SEDA)-based approach to achieve this?*
3) *Is it possible to design HDFS with a hybrid architecture to take advantage of the heterogeneous storage devices on HPC clusters?*
4) *Can we propose advanced acceleration techniques to adapt HDFS with in-memory and heterogeneous storage for iterative benchmarks and applications?*
5) *How can we take advantage of high-performance burst buffer system to accelerate I/O performance of Big Data applications on HPC clusters? Can key-value store such as Memcached be used for this? What are the challenges to integrate Hadoop with Lustre through this burst buffer?*
6) *Can we re-design HDFS to leverage the byte-addressability of NVM?*
7) *Can efficient data access strategies be proposed to accelerate Big Data analytics?*

II. Major Contributions and Results

The designs proposed in this thesis address the above-mentioned challenges and have been incorporated in the RDMA for Apache Hadoop software package released under the HiBD project [3]. Figure 2 shows the overall scope of this thesis. In this section, the designs and corresponding results are discussed in detail.

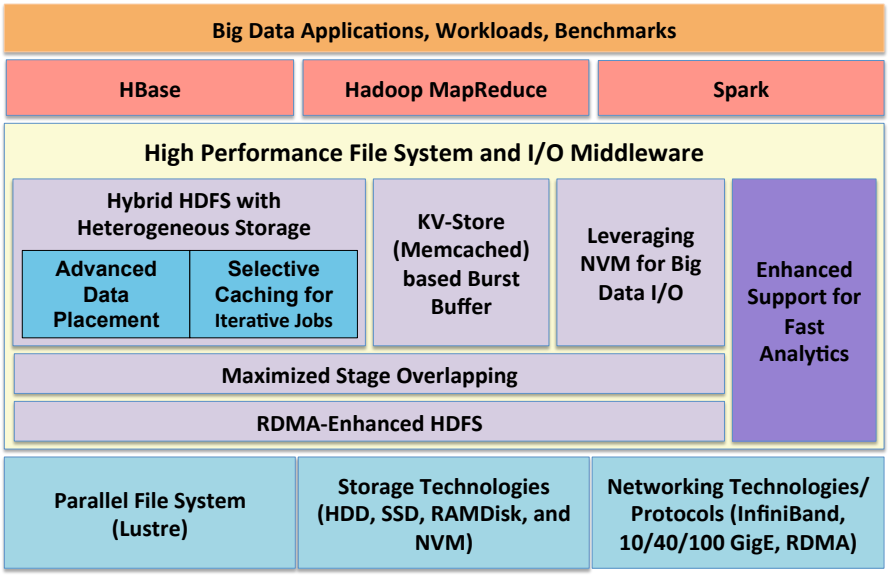
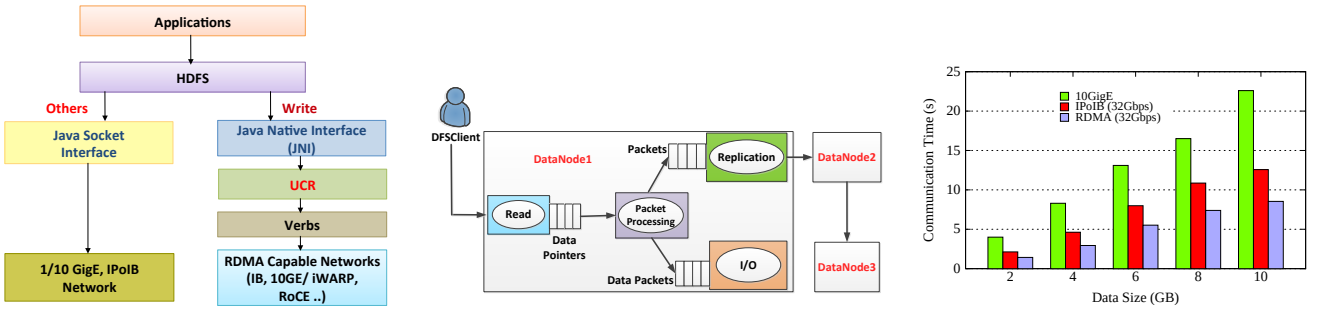


Fig. 2: The Proposed Research Framework

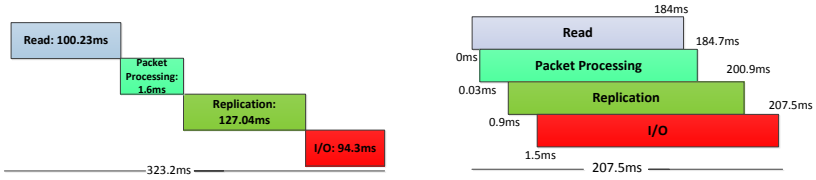
1) **RDMA-Enhanced HDFS with Maximized Overlapping [7, 8]**: HDFS cannot leverage the high performance communication features of HPC clusters. High performance networks such as InfiniBand [5] provide low latency and high throughput data transmission. Over the past decade, scientific and parallel computing domains, with the Message Passing Interface (MPI) as the underlying basis for most applications, have made extensive usage of these advanced networks. Implementations of MPI, such as MVAPICH2 [12], achieve low one-way latencies in the range of 1-2μs. On the other hand, even the best implementation of sockets on InfiniBand achieves 20-25μs one-way latency [4]. HDFS is communication intensive due to its distributed nature. All existing communication protocols [1] of HDFS are layered on top of TCP/IP. Due to the byte-stream communication nature of TCP/IP, multiple data copies are required, which results in poor performance in terms of both latency and throughput. Consequently, even though the underlying system is equipped with high performance interconnects such as InfiniBand, HDFS cannot fully utilize the hardware capability and obtain peak performance. To address this drawback, in this work, an RDMA-Enhanced HDFS design has been proposed in which HDFS write and replication (reads are mostly node-local) go over RDMA while all other HDFS operations go over Java-Socket. The JNI layer bridges Java-based HDFS with communication library written in native C. Figure 3(a) illustrates our design. The use of JNI direct buffer ensures zero-copy data transfer in our design.
During HDFS `Write`, a data block is transferred as packets from clients to DataNodes. Each packet goes through processing and replication; finally, it is stored inside the DataNode. The vanilla HDFS adopts the One-Block-One-Thread (OBOT) architecture to process data sequentially. The OBOT design is a good trade-off between simplicity and performance for default Hadoop running over low-speed interconnect clusters. Even though this design can support task/block-level parallelism, there is no overlapping among different packets belonging to the same block. Moreover, with RDMA-based communication, the data transfer time is significantly reduced which results in higher message rate in the



(a) RDMA-Enhanced HDFS Design (b) Maximized Overlapping in RDMA-Enhanced HDFS (c) Communication time in RDMA-Enhanced HDFS

Fig. 3: Design and performance of RDMA-Enhanced HDFS with maximized overlapping

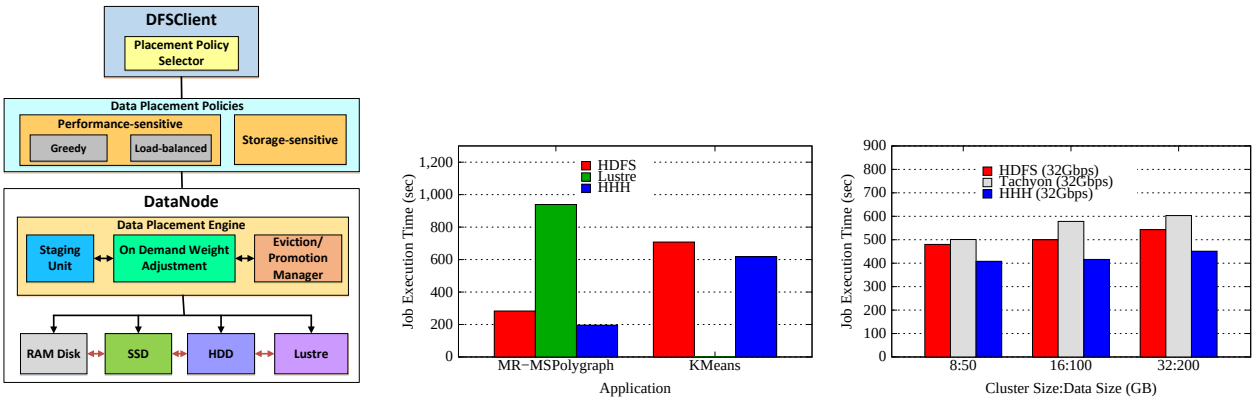
DataNode side. Due to the sequential data processing feature in the OBOT architecture, the incoming data packets should wait for the I/O stage of the previous packet to be completed before they are read and processed. It is, therefore, critical to design RDMA-Enhanced HDFS with a higher throughput architecture. For this, the Staged Event-Driven Architecture (SEDA)-based approach is adopted to maximize overlapping among different stages as depicted in Figure 3(b). With SEDA, the operations of *Read*, *Packet Processing*, *Replication*, and *I/O* stages are realized by separate pool of threads connected through queues. As shown in Figure 3(c), the RDMA-Enhanced design reduces the communication time of HDFS write by up to 30% over IPoIB (32Gbps) and 56% over 10GigE. Figure 4 further demonstrates that the overlapping introduced by the proposed design reduces the execution time of HDFS write by 35.8%.



(a) Sequential operations in default architecture (b) Overlapped stages in RDMA-enhanced HDFS

Fig. 4: Breakdown time for 64MB block write

2) **Hybrid HDFS with Heterogeneous Storage (HHH) [6, 10]**: HDFS cannot efficiently utilize the heterogeneous storage devices available on HPC clusters. The limitations come from the existing placement policies and ignorance of data usage patterns. HDFS in its default architecture, does not make use of Lustre and uses the local storage devices in Round-Robin fashion for data placement.



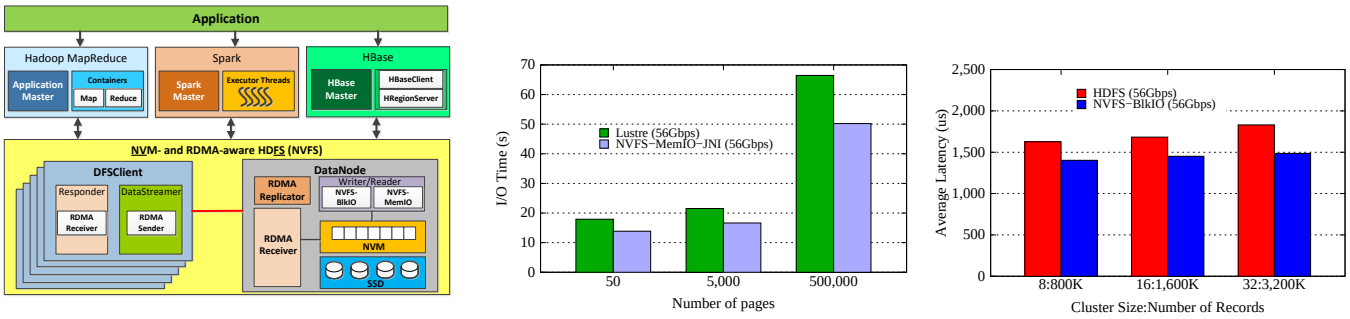
(a) HHH Architecture (b) Evaluation of HHH with MapReduce Applications (c) Evaluation of HHH with Spark Workload (TeraSort)

Fig. 5: HHH Design and Evaluations

But in the presence of heterogeneous storage devices, Round-Robin cannot always guarantee optimal performance. Therefore, in this work enhanced data placement policies for HDFS have been proposed, that can efficiently utilize the available storage devices. The storage-sensitive placement policy that utilizes Lustre for fault-tolerance, thereby, avoiding HDFS replication, can reduce the local storage requirement by 66% while ensuring better data-locality over Lustre. Figure 5(a) demonstrates the architecture of the proposed design. This design also evicts/promotes data to appropriate

storage layers based on their access patterns. In this way, I/O bottlenecks are reduced for data-intensive applications. Iterative applications, on the other hand, suffer from the huge I/O bottlenecks of HDFS due to persisting the data from each intermediate iteration. Such applications also generate many different types of data. But blindly storing all the data to high performance storage is neither economical nor necessary for improving application performance. Therefore, this work proposes to selectively cache the data from the intermediate iterations. As observed from Figure 5(b) and Figure 5(c), HHH design can significantly improve the performance of different types (batch and iterative (Evaluating KMeans over Lustre was out of scope) of MapReduce and Spark workloads. In order to further facilitate parallel file system access, a key-value store-based burst buffer has also been designed to integrate Hadoop with Lustre, so that applications do not suffer from the bandwidth bottlenecks of shared file system access [11].

3) **High Performance Design of HDFS with NVM and RDMA (NVFS) [9]**: For performance-sensitive applications, in-memory storage is being increasingly used for HDFS on HPC systems. Even though HPC clusters are equipped with large memory per compute node, using this memory for storage can lead to degraded computation performance due to competition for physical memory between computation and I/O. RDMA-based communication in HDFS also need large amount of memory in the presence of many concurrent clients and replicators. Persistence is also challenging for in-memory data placement. Therefore, this work presents an NVM-based HDFS design to leverage the byte-addressability of NVM for RDMA-based communication. In NVFS, HDFS I/O has been re-designed with memory semantics to fully exploit the byte-addressability. This work also presents cost-effective acceleration techniques for HBase and Spark to utilize the NVM-based design of HDFS by storing only the HBase Write Ahead Logs and Spark job outputs to NVM, respectively. Further enhancements to use the NVFS design as a burst buffer for running Spark jobs on top of parallel file systems like Lustre have also been proposed. Figure 6(a) illustrates the design of NVFS. As observed from Figure 6(b) and Figure 6(c), NVFS can significantly improve the performances of Spark and HBase.



(a) NVFS Design Overview (b) Evaluation of NVFS with Spark PageRank (c) Evaluation of NVFS with HBase Insert

Fig. 6: NVFS Design and Evaluations

In the future, efficient data access strategies will be proposed to accelerate Big Data Analytics on HPC platforms.

REFERENCES

[1] "The Apache Hadoop Project," http://hadoop.apache.org/.

[2] Gordon at San Diego Supercomputer Center, http://www.sdsc.edu/us/resources/gordon/.

[3] HiBD, http://hibd.cse.ohio-state.edu/.

[4] J. Huang, X. Ouyang, J. Jose, M. W. Rahman, H. Wang, M. Luo, H. Subramoni, C. Murthy, and D. K. Panda, "High-Performance Design of HBase with RDMA over InfiniBand," in *The Proceedings of IEEE 26th Intl. Parallel and Distributed Processing Symposium (IPDPS)*, Shanghai, China, May 2012.

[5] Infiniband Trade Association, http://www.infinibandta.org.

[6] N. S. Islam, X. Lu, M. W. Rahman, D. Shankar, and D. K. Panda, "Triple-H: A Hybrid Approach to Accelerate HDFS on HPC Clusters with Heterogeneous Storage Architecture," in *15th IEEE/ACM Intl. Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2015.

[7] N. S. Islam, X. Lu, M. W. Rahman, and D. K. Panda, "SOR-HDFS: A SEDA-based Approach to Maximize Overlapping in RDMA-Enhanced HDFS," to appear in 23rd Intl. ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC), 2014.

[8] N. S. Islam, M. W. Rahman, J. Jose, R. Rajachandrasekar, H. Wang, H. Subramoni, C. Murthy, and D. K. Panda, "High Performance RDMA-based Design of HDFS over InfiniBand," in *Intl. Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2012.

[9] N. S. Islam, M. W. Rahman, X. Lu, and D. K. Panda, "High Performance Design for HDFS with Byte-Addressability of NVM and RDMA," in *Intl. Conference on Supercomputing (ICS)*, 2016.

[10] N. S. Islam, M. W. Rahman, X. Lu, D. Shankar, and D. K. Panda, "Performance Characterization and Acceleration of In-Memory File Systems for Hadoop and Spark Applications on HPC Clusters," in *2015 IEEE International Conference on Big Data (IEEE BigData)*, 2015.

[11] N. S. Islam, D. Shankar, X. Lu, M. W. Rahman, and D. K. Panda, "Accelerating I/O Performance of Big Data Analytics on HPC Clusters through RDMA-based Key-Value Store," in *Intl. Conference on Parallel Processing (ICPP)*, 2015.

[12] MVAPICH2: High Performance MPI over InfiniBand and iWARP, http://mvapich.cse.ohio-state.edu/.

[13] T. Sterling, E. Lusk, and W. Gropp, "Beowulf Cluster Computing with Linux," in *MIT Press*, 2003.