MapReduce over Lustre: Can RDMA-Based Approach Benefit? *

Md. Wasi-ur-Rahman, Xiaoyi Lu, Nusrat Sharmin Islam,
Raghunath Rajachandrasekar, and Dhabaleswar K. (DK) Panda

Department of Computer Science and Engineering,
The Ohio State University
{rahmanmd,luxi,islamn,rajachan,panda}@cse.ohio-state.edu

Abstract. Recently, MapReduce is getting deployed over many High Performance Computing (HPC) clusters. Different studies reveal that by leveraging the benefits of high-performance interconnects like InfiniBand in these clusters, faster MapReduce job execution can be obtained by using additional performance enhancing features. Although RDMA-enhanced MapReduce has been proven to provide faster solutions over Hadoop distributed file system, efficiencies over parallel file systems used in HPC clusters are yet to be discovered. In this paper, we present a complete methodology for evaluating MapReduce over Lustre file system to provide insights about the interactions of different system components in HPC clusters. Our performance evaluation shows that RDMA-enhanced MapReduce can achieve significant benefits in terms of execution time (49% in a 128-node HPC cluster) and resource utilization, compared to the default architecture. To the best of our knowledge, this is the first attempt to evaluate RDMA-enhanced MapReduce over Lustre file system on HPC clusters.

Keywords: MapReduce, RDMA, Lustre, HPC Clusters.

1 Introduction

The explosive growth of 'Big Data' has caused many industrial firms to adopt HPC technologies to meet the requirements of huge amount of data to be processed and stored. According to the IDC study [6] in 2013, 67% of HPC sites were running High-Performance Data Analysis (HPDA) workloads. Hadoop MapReduce [21] and Hadoop Distributed File System (HDFS) [16] are increasingly being used on modern HPC clusters [17,4] to process HPDA workloads.

The default Hadoop design mainly focuses on the commodity servers which are typically equipped with low-bandwidth interconnects. These clusters often have multiple large-capacity local HDDs to achieve better data-locality for MapReduce jobs. In contrast, modern HPC clusters [17,4] have quite different execution environments, where

* This research is supported in part by National Science Foundation grants #OCI-1148371, #CCF-1213084 and #CNS-1347189. It used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575.

F. Silva et al. (Eds.): Euro-Par 2014, LNCS 8632, pp. 644–655, 2014.

high-speed interconnects, like InfiniBand, 10 Gigabit Ethernet (10 GigE), and high performance but smaller capacity local disks are commonly used. In addition, a global file system, like Lustre [24], is often shared by all the compute nodes to meet the storage requirements of HPC applications. If we directly run default Hadoop on HPC clusters, it is hard to achieve optimal performance; because, recent studies [14,23,8,13,10,2,5] have shown that default Hadoop components can not leverage HPC cluster features, like Remote Direct Memory Access (RDMA) enabled high performance interconnects, high-throughput and large capacity parallel file systems, etc. efficiently. InfiniBand is the most popular RDMA-enabled high-performance interconnect in TOP500 [22], while Lustre [24] is widely deployed on modern HPC clusters.

1.1 Motivation and Related Studies

The use of Lustre, in particular, with the MapReduce architecture has attracted significant attention within the Big Data community. The evaluations in [9,2,15] argue for the use of Lustre as the back-end file system for Apache Hadoop MapReduce in HPC clusters. On the other hand, recent studies [14,13] show that by leveraging the benefits of RDMA, the overall performance of Hadoop MapReduce can be significantly improved with many additional design features. Both default and RDMA-enhanced designs of MapReduce have been well studied and evaluated with default and RDMA-enhanced HDFS designs.

Table 1. Existing Performance Studies on MapReduce Designs

	Apache HDFS	RDMA HDFS	Lustre
Apache MapReduce	[21,16]	[8]	[2,15,9]
RDMA MapReduce	[14,23,13]	[11]	N/A (this paper)

Table 1 summarizes the existing studies on different combinations of MapReduce designs with different file systems. As shown here, the benefits of RDMA-enhanced MapReduce over Lustre are not yet discovered. In this regard, an obvious question arises: *Can RDMA-based Approaches Benefit MapReduce over Lustre?*

1.2 Contributions

This paper addresses this issue by comparing our RDMA-enhanced MapReduce [14] solution with default MapReduce over Lustre. The primary contributions of this paper are as follows:

1. A demonstration of the potential of RDMA-enhanced MapReduce over Lustre deployments on leadership-class HPC systems,
2. A comprehensive methodology, to evaluate MapReduce solutions over parallel file systems provisioned on such HPC systems, and to understand the behavior of Hadoop on HPC resources, and
3. A thorough evaluation of both default and RDMA-enhanced designs of MapReduce, to give insights into the benefits of RDMA in terms of scalability, performance, and resource utilization efficiency.

In our performance evaluations, we observe 49% benefit in job execution time for the Sort benchmark with an increasing cluster size of up to 128 nodes in Cluster TACC-Stampede. On SDSC-Gordon, a 43% benefit in job execution time is observed in comparison to the default architecture, on evaluations up to 64 nodes. To the best of our knowledge, this is the first paper to show the benefits of RDMA-enhanced MapReduce over Lustre in production HPC clusters.

2 Evaluation Methodology

In this section, we discuss our evaluation methodology in detail.

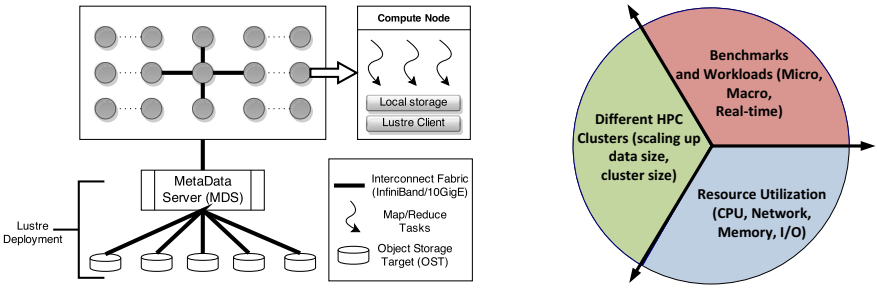
2.1 Evaluation Platforms

Most of the modern HPC clusters follow a hybrid topological solution of traditional Beowulf architecture [19,20] with separate I/O service nodes. The architecture of these clusters opens the possibility of keeping lean compute nodes with lightweight operating system and limited storage capacity [3], connected to a sub-cluster of dedicated I/O nodes with enhanced parallel file systems, such as Lustre, to provide fast and scalable solutions. Figure 1(a) shows such a deployment where dedicated I/O nodes are reserved as Metadata Servers (MDS) and Object Storage Targets (OST) for Lustre, that are connected to the client compute nodes through high performance interconnects, typically InfiniBand or 10 GigE. Each of the compute nodes has small local storage as well as Lustre client to read/write data to Lustre. When a MapReduce framework is configured to run in such a cluster, TaskTrackers and Map/Reduce Tasks are launched in compute nodes. These processes use the local storage for temporary data and Lustre for persistent storage. Typically, a MapReduce application can be CPU-, I/O-, and/or network-bound as it goes through different stages involving any or a combination of these operations. Thus, we propose an evaluation methodology that considers application behavior for all system and configuration settings.

2.2 Dimensions in Methodology

To propose an evaluation methodology for such deployment, we emphasize three broad dimensions, shown in Figure 1(b).

Different HPC Clusters: Popular clusters used in the HPC community vary based on the number of system, network, and I/O resources available which brings variations in application performance behavior. Also, difference in configuration and problem size add further variability on performance for MapReduce jobs over any file system. In this paper, we choose three clusters with MapReduce over Lustre deployment. Among these, TACC-Stampede [17] is one of the largest supercomputing system based on 6,400+ Dell PowerEdge server nodes. According to TOP500 [22] list in November 2013, this cluster is listed as the 7th fastest supercomputer worldwide with a delivered performance of 5,168.1 TFlops. SDSC-Gordon [4], ranked 129th in the same list, is another large HPC cluster that we use for our evaluation. We choose these clusters to bring enough variations in our experimental setup. For example, these two clusters differ in Lustre



(a) MapReduce over Lustre on modern HPC clusters (b) Different dimensions in methodology

Fig. 1. Evaluation basis and dimensions

interconnect (10 GigE in SDSC-Gordon and InfiniBand FDR in TACC-Stampede) as well as interconnect for compute nodes (InfiniBand QDR vs FDR). The third cluster we choose has InfiniBand QDR as the interconnect for both Lustre and compute nodes.

Benchmarks and Workloads: We select a set of benchmarks and workloads to facilitate variations in workload characteristics. We categorize the benchmarks into three different types: micro-benchmarks, macro-benchmarks, and real-world workloads. For micro-benchmark, we select the Sort, as this is one of the simplest and most popular MapReduce benchmarks with minimal user-defined `map()` and `reduce()` functionality. For macro-benchmarks, we select five different benchmarks from Purdue MapReduce Benchmark Suite (PUMA) [12,1] based on the benchmark characteristics. In particular, we consider the ratio of the data volume that gets shuffled to the data volume in computation and choose two shuffle-intensive (Adjacency List and Self Join) and two compute-intensive (Word Count and Inverted Index) benchmarks to introduce enough variation in our evaluation. However, we also pick Sequence Count benchmark that qualifies for both as it has both computation and shuffle over large volume of data. We also choose the Statistical Workload Injector for MapReduce (SWIM) [18] that provides real-world workload from production clusters in Facebook. This workload consists of many short-duration MapReduce jobs that run one after another in an overlapped manner to mimic the workload characteristics in the data center environment.

Resource Utilization: Resource utilization determines the ability of a framework to provide fast and scalable solutions. In this purpose, we select the following parameters.

CPU Usage: For most MapReduce applications, `map()` and `reduce()` phases consume most of the CPU cycles, as these are user-defined functions that directly operate on the data. To provide more CPU cycles to these functions, the underlying framework must keep the CPU free most of the time during its execution. Hence, it is critical to understand CPU utilization patterns for any MapReduce framework.

Memory Usage: Most of the modern HPC clusters provide large amount of memory on each machine that can be utilized during execution of MapReduce or any Big Data applications to achieve faster job execution throughput. Although default MapReduce relies heavily on both disk and memory usage, faster solutions are possible by utilizing memory space more than disk. In this perspective, this parameter is crucial.

Network Throughput: With traditional network interfaces, the shuffle phase acts as the bottleneck in the job execution pipeline due to their limited network bandwidth. However, due to the presence of modern high performance interconnects in the HPC clusters, the bottleneck in shuffle phase turns into the question of how efficiently the available network bandwidth can be utilized by the MapReduce framework so that all the other phases can benefit from faster data communication. With the profiling of this parameter, differences of the underlying protocol stacks can also be discovered which may help to realize the better network stack for such applications in HPC clusters.

I/O Usage: In Big Data applications, I/O usage is fundamental as the input and output of the problem space are usually provided from and written to underlying file systems, the storage of which is backed by HDDs, SSDs, or both. However, abundant use of disks may cause slowness in the pipeline which reduces the performance of MapReduce applications. In this perspective, I/O usage parameter is useful to visualize whether a particular MapReduce framework can observe benefits by reducing significant number of I/O operations. For any MapReduce framework, the initial read and final write phases are obligatory. Thus, only the local I/O operations during shuffle phase are those that can be reduced to have an impact in the overall job execution. In our evaluation, we measure the IOPS (I/O per second) for the local disk.

2.3 Evaluation Methods

To measure different parameters mentioned in Section 2.2, we use Linux performance monitoring tool, `sar`, provided as a part of the `sysstat` package. `sar` can be used to measure real-time data for CPU, memory, and I/O usage. We measure all of the metrics on the entire cluster to monitor all the concurrent tasks for overall performance. The sampling rate we use is two seconds. For reporting CPU usage, we use an arithmetic average over all CPUs' usage obtained from different machines in the cluster. For memory and I/O usage, similar methods are followed by measuring the parameters, free memory (kbmemfree) and transaction per second (tps), respectively. For network throughput measurement, we profile the shuffle stage to report the amount of data transfer at each point of time. We sum the total data transfer at the second granularity and average over all data transfers in the cluster. This resembles the overall data transfer capability in the shuffle phase of the corresponding framework.

3 Performance Evaluation

In this section, we discuss experimental setups and detailed performance evaluations.

Table 2 summarizes our three clusters' configurations. The Lustre deployments at OSU and TACC-Stampede are accessible through the InfiniBand interconnect, while that at SDSC-Gordon uses a 10 GigE transport. We used hadoop-0.20.2 and Java 1.7 for our experiments. As InfiniBand software stacks provide a driver for implementing the IP layer, we evaluate the default MapReduce over this layer. This is indicated as "IPoIB" (IP-over-InfiniBand) in the subsequent graphs. The "RDMA" legends in the graphs represent RDMA-enhanced MapReduce architecture [14] which uses native IB for communication. QDR and FDR are mentioned as 32Gbps and 56Gbps, respectively.

Table 2. Experimental setups used in this paper

Cluster	**OSU**	**SDSC-Gordon**	**TACC-Stampede**
Nodes (cores)	25 (200)	65 (1040)	129 (2064)
Processor	Intel Xeon E5640 dual quad-core (2.67 GHz)	Intel EM64T Xeon E5 dual octa-core (2.7 GHz)	Intel Sandy Bridge E5-2680 dual octa-core (2.6 GHz)
Memory	12/24 GB per node	64 GB per node	32 GB per node
Local disk	single 160 GB HDD per node	single 80 GB HDD per node	single 80 GB HDD per node
Lustre	12 TB	4 PB	14 PB
OS	Red Hat Enterprise Linux Server 6.4	CentOS 6.4 (Final)	CentOS 6.3 (Final)
Interconnect (compute nodes)	InfiniBand QDR (32Gbps)	InfiniBand QDR (32Gbps)	InfiniBand FDR (56Gbps)
Interconnect (Lustre)	InfiniBand QDR (32Gbps)	10GigE	InfiniBand FDR (56Gbps)

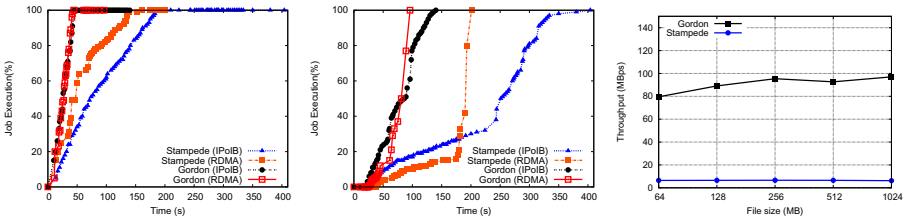
3.1 Tuning of Lustre Stripe Size

The total number of launched map tasks in a MapReduce job execution depends on the file system block size as each map reads one block of data. Tuning the block size can get a good trade-off point between I/O and parallel task execution. For MapReduce over Lustre, the Lustre stripe size is set to the block size to ensure that each block resides in single OST, rather than a multiple number of OSTs. We use the Sort micro-benchmark with different stripe sizes for both default architecture and RDMA-enhanced design. For these experiments, we have used a cluster size of four with a data size of 20 GB. A stripe size of 64 MB is proved to be optimum for IPoIB in Cluster OSU. However, for RDMA-enhanced design, 256 MB stripe size obtains the best performance in terms of job completion. For SDSC-Gordon, both IPoIB and RDMA have an optimum stripe size of 256 MB. In Cluster TACC-Stampede, stripe size is tuned to a value of 128 MB for IPoIB. For RDMA, performance is mostly similar across different stripe sizes starting from 64 MB to 512 MB. We pick 256 MB as the optimum stripe size for RDMA-enhanced design in TACC-Stampede. For the remaining experiments, we have used the optimum Lustre stripe size obtained from these tunings.

3.2 Comparison of Progress in Different Phases

We measure the execution progress with respect to time for map and reduce phases and compare the results in Figure 2. We show these results for Cluster TACC-Stampede and SDSC-Gordon.

Figure 2(a) shows the map execution progress for 20 GB Sort in a cluster size of four. Here, we can see that, map phase in RDMA-enhanced design finishes a little early compared to that of default architecture. This is because RDMA-based shuffle allows more CPU cycles to be available for `map()` execution. Also, the difference of map progress between Cluster TACC-Stampede and SDSC-Gordon occurs because of the difference in local disk write throughput between these two clusters (shown later in Figure 8(c) and Figure 8(b)). For the reduce progress shown in Figure 2(b), we see that the RDMA-enhanced approach is much faster in progress in both the clusters compared to the default architecture because of its design features. However, the difference in progress between the two clusters, TACC-Stampede and SDSC-Gordon, is primarily due to the difference in the underlying Lustre write throughput. We perform write experiments with IOzone [7] benchmark to measure throughput of Lustre file system in



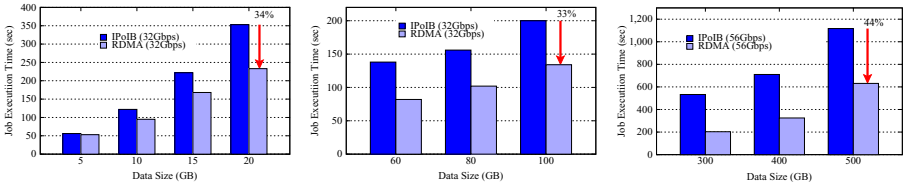
(a) Map progress (b) Reduce progress (c) Lustre throughput

Fig. 2. Map and reduce phase progress comparison in different clusters

both clusters. As shown in Figure 2(c), SDSC-Gordon Lustre deployment has much improved throughput (90 MBps average) compared to that (6.5 MBps average) of TACC-Stampede.

3.3 Evaluation of Micro-benchmark

In these experiments, we measure the job execution time for both architectures and compare them with varying cluster and data sizes.

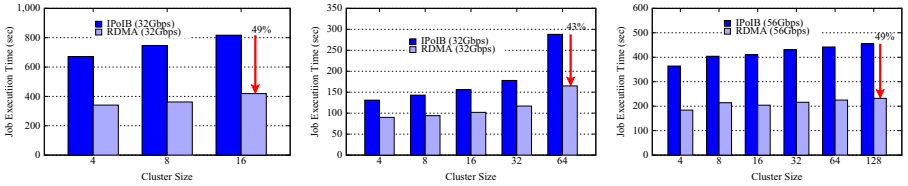


(a) OSU (4 nodes) (b) SDSC-Gordon (16 nodes) (c) TACC-Stampede (32 nodes)

Fig. 3. Sort benchmark evaluation with variation in data size

Figure 3 presents the Sort benchmark evaluation on different clusters based on data size variation. In Figure 3(a), we present the job execution times in Cluster OSU varying the data size from 5 GB to 20 GB. For increased data size, we observe a trend of increase in improvement for RDMA-enhanced design. For 20 GB data size, it has a performance benefit of 34% compared to IPoIB. For similar experiments in SDSC-Gordon (Figure 3(b)), we vary data size from 60 GB to 100 GB. For this experiment, the performance benefit of RDMA-enhanced approach is 33% for 100 GB Sort. In TACC-Stampede (Figure 3(c)), we vary the data size from 300 GB to 500 GB. Here, we observe a benefit of 44% for 500 GB Sort.

We also conduct experiments with simultaneous variations in both cluster and data sizes. We present these results in Figure 4. Here, in Figure 4(a), we increase cluster size from 4 to 16 with a data size increase of 20 to 80 GB for Cluster OSU. RDMA-enhanced approach observes a benefit of 49% here for cluster size 16. For similar experiments in SDSC-Gordon, we vary cluster size from 4 to 64 with data size increase of



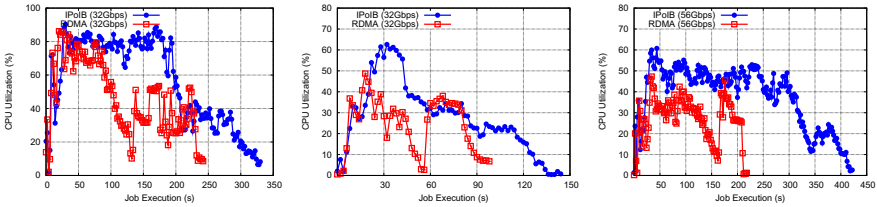
(a) OSU (b) SDSC-Gordon (c) TACC-Stampede

Fig. 4. Sort benchmark evaluation with variation in cluster and data size

up to 320 GB and observe a performance benefit of 43% for the largest cluster and data size. For TACC-Stampede (Figure 4(c)), we vary the cluster size up to 128 nodes with 640 GB data size and achieve a performance benefit of 49%. In all these experiments, we observe a trend of increase in performance benefit for RDMA-enhanced design as we scale up both the cluster size and the data size. For TACC-Stampede, the benefit is more compared to SDSC-Gordon because of the InfiniBand FDR interconnect.

3.4 Evaluation of Resource Utilization

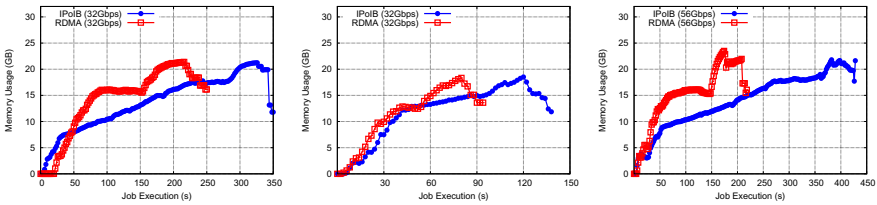
In this section, we use profiling analysis to find out different system resource utilization. We use Sort benchmark to evaluate all the metrics in a cluster size of four.



(a) OSU (b) SDSC-Gordon (c) TACC-Stampede

Fig. 5. Profiling CPU usage

CPU: We present the CPU usage for different clusters in Figure 5. Here, X-axis represents the job execution in seconds and Y-axis represents the average CPU usage at each point of time during job execution. We profile CPU idle% on each machine in the cluster and average across all the CPUs. As shown in Figure 5, in all the clusters, default architecture is more CPU hungry compared to RDMA-enhanced approach. The major reason behind this observation is that, RDMA-enhanced communication does not require remote end's CPU to perform data transmission. Besides, during the early stage when all maps are running, the default architecture tries to shuffle the map output data as much as possible, keeping the ReduceTasks busy and taking longer CPU cycles. However, in RDMA-enhanced design, during this phase, the ReduceTasks are shuffling only a small portion of map output data to build up the Priority Queue and thus takes



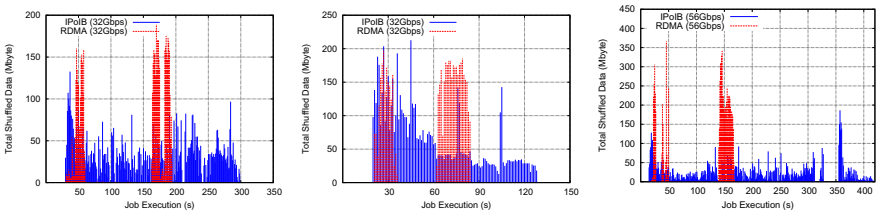
(a) OSU (b) SDSC-Gordon (c) TACC-Stampede

Fig. 6. Profiling memory usage

less CPU cycle. The average reductions in CPU usage for the three clusters in the order of Figure 5 are 14%, 13%, and 28%, respectively.

Memory: We observe memory usage for both the architectures and present the comparisons in Figure 6. The memory requirements during the entire job execution are similar for both architectures. However, RDMA-enhanced design utilizes free memory better compared to the default architecture. From the time of initialization, the RDMA-enhanced design capitalizes on using the available free memory space to achieve faster job completion and thus reduces the duration of memory consumption for Cluster OSU, SDSC-Gordon, and TACC-Stampede by 27%, 35%, and 57% respectively.

Network: In Figure 7, we present the comparison of shuffled data transfer during the job execution. The X-axis represents the job execution progress in seconds and the Y-axis represents the average data transfer from a single TaskTracker to all ReduceTasks at each point of time during the job execution. It clearly shows that the default architecture over IPoIB can not take advantage of network bandwidth as it transfers data during the entire job execution process with an average throughput of only 24.75 Mbytes/sec in Cluster TACC-Stampede (shown in Figure 7(c)). This also demonstrates the fact that the pipeline efficiency in the default architecture is not good enough to handle large amount of data shuffle at once.

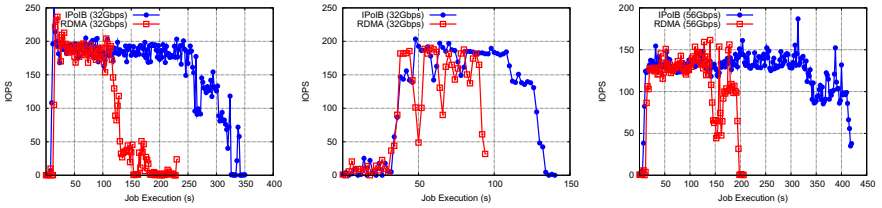


(a) OSU (IB QDR) (b) SDSC-Gordon (IB QDR) (c) TACC-Stampede (IB FDR)

Fig. 7. Profiling network throughput

However, at the early stage of the job execution, ReduceTasks in RDMA-enhanced design build the Priority Queue with the least amount of data transferred from each map location. As soon as the map phase completes, it utilizes the network more and

achieves an average throughput of 77 Mbytes/sec, thus observing 211% improvement over default architecture. However, the average throughput after the map phase is as high as 197 Mbytes/sec that clearly reflects how efficiently this architecture utilizes network bandwidth as soon as it gets the opportunity. This also states the fact that, in the RDMA-enhanced design, the pipeline efficiency after the map phase is good enough to consider a high network throughput. For Cluster OSU (Figure 7(a)) and SDSC-Gordon (Figure 7(b)), the absolute values of achievable network bandwidth for both the architectures are less due to the data rate of the network cards (QDR). However, RDMA-enhanced design still observes an improvement in average network throughput of 29% in both of these clusters.



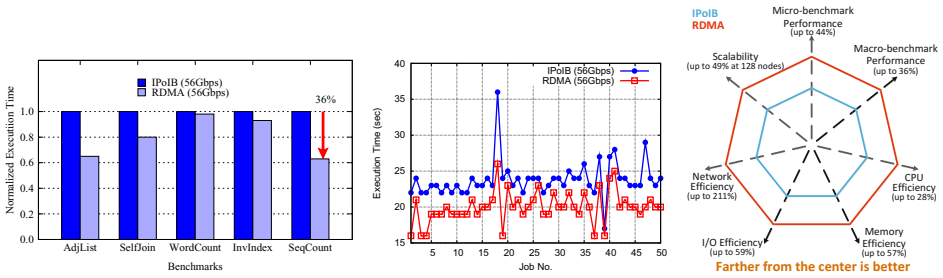
(a) OSU (b) SDSC-Gordon (c) TACC-Stampede

Fig. 8. Profiling I/O operations in local disk

Disk I/O: Figure 8 presents the profiling of local I/O operations. We present the I/O operations per second (IOPS) against job execution time for both the architectures. Here, we can see that, because of the in-memory merge operations, RDMA-enhanced design reduces IOPS to almost zero as soon as map phase completes. The reductions in IOPS for Clusters OSU, SDSC-Gordon, and TACC-Stampede are 59%, 43%, and 58%, respectively. The reason for difference in these values is due to the fact that the default architecture uses local file system for merge operation if the available memory space is not enough. So, depending on the local memory space and disk throughput, the IOPS value can vary in different clusters.

3.5 Evaluation of Macro-Benchmarks

For space limitation, we present macro-benchmarks performance comparisons in Cluster TACC-Stampede only. We use a cluster size of 32 for PUMA [12] benchmarks. As shown in Figure 9(a), we observe 35% improvement for Adjacency List with a data size of 30 GB. For Sequence Count, the benefit is 36% for 80 GB data size. With compute-intensive workloads, such as Word Count and Inverted Index, the total shuffled data volume is not significant [1]. Thus, we observe less benefits in job execution time. For SWIM [18], shown in Figure 9(b), we evaluate 50 short-duration jobs for which the data is generated from real-time workloads. We observe an average benefit of 16% in terms of job execution time with a cluster size of four.



(a) PUMA [12] (b) SWIM [18] (c) Benefits summary

Fig. 9. Macro benchmark evaluation and summary

3.6 Summary

To summarize, we present a 7-axis hypothetical Figure 9(c) with each dimension representing one of the evaluation parameters. We assume that values farther from the center signifies higher benefit in performance for that parameter. As shown here, RDMA-enhanced design [14] achieves improved performance in each dimension due to its inherent design enhancements. Both designs are scalable while RDMA-enhanced design achieves better performance with scaling up in both cluster and data size.

4 Conclusion and Future Work

In this paper, we propose a methodology to evaluate MapReduce over Lustre file system in modern HPC clusters. Our performance evaluations based on this methodology show that RDMA-enhanced MapReduce can achieve significant performance benefits compared to the default architecture in every aspect of system and resource utilization. The centerpiece of our evaluation demonstrates that RDMA techniques help reduce the job execution time by 49% on 128 node cluster, in comparison to the default architecture. As part of the future work, we would like to explore more involved techniques that help improve the performance of MapReduce over HPC file systems such as Lustre.

References

1. Ahmad, F., Chakradhar, S.T., Raghunathan, A., Vijaykumar, T.N.: Tarazu: Optimizing MapReduce on Heterogeneous Clusters. In: ASPLOS (2012)
2. Castain, R.H., Kulkarni, O.: MapReduce and Lustre: Running Hadoop in a High Performance Computing Environment, `https://intel.activeevents.com/sf13/connect/sessionDetail.ww?SESSION_ID=1141`
3. Engelmann, C., Ong, H., Scott, S.L.: Middleware in modern high performance computing system architectures. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007, Part II. LNCS, vol. 4488, pp. 784–791. Springer, Heidelberg (2007)
4. Gordon at San Diego Supercomputer Center, `http://www.sdsc.edu/us/resources/gordon/`

5. Huang, J., Ouyang, X., Jose, J., Rahman, M.W., Wang, H., Luo, M., Subramoni, H., Murthy, C., Panda, D.K.: High-Performance Design of HBase with RDMA over InfiniBand. In: IPDPS, Shanghai, China (2012)
6. International Data Corporation (IDC): New IDC Worldwide HPC End-User Study Identifies Latest Trends in High Performance Computing Usage and Spending, http://www.idc.com/getdoc.jsp?containerId=prUS24409313
7. IOzone: IOzone Filesystem Benchmark, http://www.iozone.org/
8. Islam, N.S., Rahman, M.W., Jose, J., Rajachandrasekar, R., Wang, H., Subramoni, H., Murthy, C., Panda, D.K.: High Performance RDMA-based Design of HDFS over InfiniBand. In: SC (2012)
9. Kulkarni, O.: Hadoop MapReduce over Lustre, http://www.opensfs.org/wp-content/uploads/2013/04/LUG2013_Hadoop-Lustre_OmkarKulkarni.pdf
10. Lu, X., Islam, N.S., Rahman, M.W., Jose, J., Subramoni, H., Wang, H., Panda, D.K.: High-Performance Design of Hadoop RPC with RDMA over InfiniBand. In: ICPP, France (2013)
11. OSU NBC Lab: RDMA for Apache Hadoop: High-Performance Design of Apache Hadoop over RDMA-enabled Interconnects, http://hadoop-rdma.cse.ohio-state.edu
12. Purdue MapReduce Benchmarks Suite (PUMA), http://web.ics.purdue.edu/
13. Rahman, M.W., Islam, N.S., Lu, X., Jose, J., Subramoni, H., Wang, H., Panda, D.K.: High-Performance RDMA-based Design of Hadoop MapReduce over InfiniBand. In: HPDIC, in Conjunction with IPDPS, Boston, MA (2013)
14. Rahman, M.W., Lu, X., Islam, N.S., Panda, D.K.: HOMR: A Hybrid Approach to Exploit Maximum Overlapping in MapReduce over High Performance Interconnects. In: ICS, Munich, Germany (2014)
15. Rutman, N.: Map/Reduce on Lustre, http://www.xyratex.com/sites/default/files/Xyratex_white_paper_MapReduce_1-4.pdf
16. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The Hadoop Distributed File System. In: MSST, Incline Village, Nevada (2010)
17. Stampede at TACC, http://www.tacc.utexas.edu/resources/hpc/stampede
18. Statistical Workload Injector for MapReduce, https://github.com/SWIMProjectUCB
19. Sterling, T., Lusk, E., Gropp, W.: Beowulf Cluster Computing with Linux. MIT Press, Cambridge (2003)
20. Sterling, T.L., Salmon, J., Becker, D.J., Savarese, D.F.: How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters. MIT Press, MA (1999)
21. The Apache Software Foundation: The Apache Hadoop Project, http://hadoop.apache.org/
22. Top500 Supercomputing System,http://www.top500.org
23. Wang, Y., Que, X., Yu, W., Goldenberg, D., Sehgal, D.: Hadoop Acceleration through Network Levitated Merge. In: SC, Seattle, WA (2011)
24. Xyratex: Lustre, http://wiki.lustre.org/index.php/Main_Page