Assessing the Performance Impact of High-Speed Interconnects on MapReduce *

Yandong Wang, Yizheng Jiao, Cong Xu, Xiaobing Li, Teng Wang, Xinyu Que,
Cristian Cira, Bin Wang, Zhuo Liu, Bliss Bailey, Weikuan Yu

Department of Computer Science
Auburn University
{wangyd,yzj0018,congxu,xbli,tzw0019,xque,
cmc0031,bwang,zhuoliu,bailebn,wkyu}@auburn.edu

Abstract. There is a lack of comprehensive examination of the performance impact of high-speed interconnects on MapReduce programs. In this work, we systematically evaluate the performance impact of two popular high-speed interconnects, 10 Gigabit Ethernet and InfiniBand, using the original Apache Hadoop and our extended Hadoop Acceleration framework. Our analysis shows that, under the Apache Hadoop, although using fast networks can efficiently accelerate the jobs with small intermediate data sizes, it is unable to maintain such advantages for jobs with large intermediate data. In contrast, Hadoop Acceleration provides better performance for jobs of a wide range of data sizes. In addition, both implementations exhibit good scalability under different networks. Hadoop Acceleration significantly reduces CPU utilization and I/O wait time of MapReduce programs.

1 Introduction

MapReduce, introduced by Google, has evolved as the backbone framework for massive-scale data analysis. Its simple yet expressive interfaces, efficient scalability, and strong fault-tolerance have attracted a growing number of organizations to build their cloud services on top of the MapReduce framework. Hadoop MapReduce [1], advanced by Apache foundation, is a popular open source implementation of MapReduce programming model. Compliant with MapReduce framewrok, Hadoop divides a job into two types of tasks, called MapTasks and ReduceTasks, and assigns them to different machines for parallel processing. Although this framework is straightforward, its intermediate data shuffling remains a critical and time-consuming operation as identified by many previous works [2, 3, 4, 5, 6]. Such data shuffling stage moves all the intermediate data generated by MapTasks to ReduceTasks, causing a significant volume of network traffics and constraining the efficiency of data analytics applications.

High-Performance interconnects, such as InfiniBand [7] and 10 Gigabit Ethernet, provide appealing solutions to relieve such pressure on the intermediate data shuffling

* This research is supported in part by an Alabama Innovation Award, an NSF grant #CNS-1059376, and a grant from Lawrence Livermore National Laboratory.

in the MapReduce frameworks. The state-of-the-art high-speed networks, such as InfiniBand, have been able to deliver up to 56Gbps bandwidth and sub-microsecond latency. Their low CPU utilization ability can spare more CPU cycles for MapReduce applications to accelerate their progress. Moreover, many fast interconnects have also offered Remote Direct Memory Access (RDMA) [8] to fully take advantages of high-performance properties of those networks. Although high-performance interconnects have been popular in the High-Performance Computing (HPC) community, the performance impact of these networks on MapReduce programs remains unclear. A growing number of cloud companies, such as Amazon [9] are planning to build their next generation clusters on top of high-performance interconnects. But there is a lack of comprehensive examination of the performance impact of these interconnects on MapReduce programs.

To fill this void, we conduct a thorough assessment of the performance of MapReduce programs on different high-performance interconnects. In particular, we investigate how InfiniBand and 10 Gigabit Ethernet (10 GigE) improve the performance of the original Hadoop and our extended Hadoop Acceleration implementation [10]. Our evaluation centers around three aspects of MapReduce clusters, including scalability, performance impact on different phases, and resource utilization. Overall, our contributions can be summarized as the following:

- For the original Apache Hadoop, using high-performance interconnects can efficiently accelerate the programs with small intermediate data size by as much as 51.5%, but provide imperceptible performance improvement for programs with large intermediate data, when compared to 1 Gigabit Ethernet (1 GigE).
- The original Hadoop exhibits good scalability under 1/10 Gigabit Ethernets and InfiniBand environments. Compared to 1 GigE, 10 GigE and InfiniBand provide better scalability in large clusters.
- Simply adopting high-performance interconnects, the original Hadoop cannot reduce the CPU utilization and remove the disk bottleneck issues in existing design of Hadoop MapReduce.
- Our Hadoop Acceleration framework exhibits comparable scalability as the Apache Hadoop. Meanwhile it is able to efficiently speed up the jobs by as much as 49.5% in both InfiniBand and 10 GigE environments. Results also show that it accelerates both the map and reduce phases of data-intensive programs.
- Our Hadoop Acceleration cuts down CPU utilization by up to 46.4% and alleviates disk contention by leveraging the advantages of fast networks.

The remainder of the paper is organized as follows. We briefly introduce the background in Section 2. We then present the comprehensive assessment results in Section 3. Finally, we provide a review of related work in Section 4 and then conclude the paper in Section 5.

2 Background

2.1 Architecture of Apache Hadoop MapReduce

Hadoop implements MapReduce framework with two categories of components: a JobTracker and many TaskTrackers. The JobTracker commands TaskTrackers to process

data in parallel through two main functions: map and reduce. In this process, the JobTracker is in charge of scheduling map tasks (MapTasks) and reduce tasks (ReduceTasks) to TaskTrackers. It also monitors their progress, collects run-time execution statistics, and handles possible faults and errors through task re-execution.

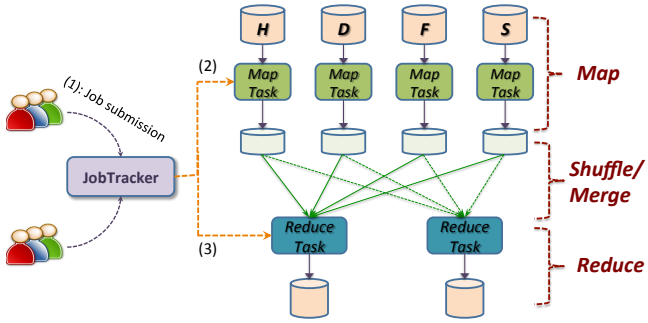


Fig. 1: An Overview of Data Processing in Hadoop MapReduce Framework

From the view of pipelined data processing, Hadoop consists of three main execution phases: map, shuffle/merge, and reduce as shown in Figure 1. In the first phase, the JobTracker selects a number of TaskTrackers and schedules them to run the map function. Each TaskTracker launches several MapTasks, one per split of data. The mapping function in a MapTask converts the original records into intermediate results, which are data records in the form of <key',val'> pairs. These new data records are stored as a MOF (Map Output File), one for every split of data. In the second phase, when MOFs are available, the JobTracker selects a set of TaskTrackers to run the ReduceTasks. TaskTrackers can spawn several concurrent ReduceTasks. Each ReduceTask starts by fetching a partition that is intended for it from a MOF (also called segment). Typically, there is one segment in each MOF for every ReduceTask. So, a ReduceTask needs to fetch such segments from all MOFs. Globally, these fetch operations lead to an all-to-all **shuffle** of data segments among all the ReduceTasks. This stage is also commonly referred as the **shuffle/merge** phase. In the third, or **reduce** phase, each ReduceTask loads and processes the merged segments using the reduce function. The final result is then stored to Hadoop Distributed File System [11].

Although the framework of Hadoop MapReduce is simple, the global all-to-all data shuffling process imposes significant pressure on the network. Therefore, it is appealing to accelerate such intermediate data shuffling via leveraging high-performance interconnects, such as InfiniBand or 10 Gigabit Ethernet (10 GigE). While, due to the higher cost of fast networks than traditional 1 Gigabit Ethernet (1 GigE), it is critical to investigate the benefits for Hadoop MapReduce to employ high performance networks.

2.2 Architecture of Hadoop-Acceleration

In order to address the issues exposed by intermediate data shuffling in Apache Hadoop, a new framework, called Hadoop Acceleration (Hadoop-A) [10], was designed on top

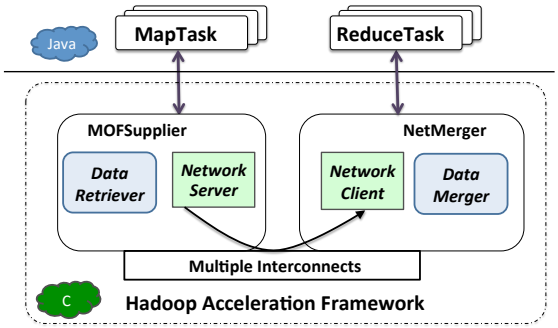


Fig. 2: Software Architecture of Hadoop-A

of Apache Hadoop to take advantage of high-speed interconnects to accelerate the data movement.

Figure 2 depicts the architecture of Hadoop Acceleration framework. Two new user-configurable plugin components, *MOFSupplier* and *NetMerger*, are introduced to leverage different protocols provided by the fast networks and enable alternative data merge algorithms. So far, the transportation protocols supported by Hadoop-A include TCP/IP and RDMA verbs. User have options to choose either Network Levitated Merge [10] or Hierarchical Merge algorithm [12] to conduct the merge process at the ReduceTask sides. Both algorithms strive to fully utilize the memory to accomplish the merge process, thus avoiding the disk I/O overhead. Both MOFSupplier and NetMerger are threaded C implementations, with all components following the object-oriented principle. We briefly describe several features of this acceleration framework without going too much into the technical details of their implementations.

User-Transparent Plugins – A primary requirement of Hadoop-A is to maintain the same programming and control interfaces for users. To this end, MOFSupplier and NetMerger plugins are designed as C processes that can be launched by TaskTrackers.

Multithreaded and Componentized MOFSupplier and Netmerger – MOFSupplier contains an network server that handles fetch requests from ReduceTasks. It also contains a data engine that manages the index and data files for all MOFs that are generated by local MapTasks. Both components are implemented with multiple threads in MOFSupplier. NetMerger is also a multithreaded program. It provides one thread for each Java ReduceTask. It also contains other threads, including a network client that fetches data partitions and a staging thread that uploads data to the Java-side ReduceTask.

2.3 Overview of High-Performance Interconnects

InfiniBand is a highly scalable interconnect technology, which can achieve low latency and high bandwidth that is up to 56Gbps. It is widely used in large data center, high performance computing systems and embedded applications, which require high speed communications. Featured by Remote Direct Memory Access (RDMA), InfiniBand transfers data directly between memory locations over network without the involvement of CPU and extra data copying. And because InfiniBand incurs very low CPU

utilization, it is ideal to carry several traffic categories, like management data and storage data, over a single connection.

10 Gigabit Ethernet (10GigE) can also attain high bandwidth but its data transmission latency is longer than InfiniBand due to the data encapsulation through TCP/IP protocol stack. RDMA is also available for 10 Gigabit Ethernet through RDMA over Converged Ethernet (RoCE). Supported by the features of RDMA, 10GigE can reduce the data transmission latency dramatically.

3 Benchmarking Study of MapReduce Programs on Different Interconnects

In this section, we report the performance of the original Apache Hadoop [1] and our Hadoop Acceleration [10], on three different interconnects.

3.1 Experimental Environment

All experiments are conducted on two environments, which are InfiniBand environment and Ethernets environment, respectively. Each environment features 23 compute nodes. All compute nodes in both clusters are identical. Each node is equipped with four 2.67GHz hex-core Intel Xeon X5650 CPUs, two Western Digital SATA 7200 RPM 500GB hard drives and 24GB memory.

In the Ethernet environment, all compute nodes connect to both a 1 Gigabit NETGEAR switch and a 10 Gigabit Voltaire switch. In the InfiniBand environment, Mellanox ConnectX-2 QDR Host Channel Adaptors are installed on each node that connect to a 108-port InfiniBand QDR switch providing up to 40 Gb/s full bisection bandwidth per port. We use the InfiniBand software stack, OpenFabrics Enterprise Distribution (OFED) [13] version 1.5.3, as released by Mellanox. Note that in the InfiniBand environment, IPoIB (an emulated implementation of TCP/IP on InfiniBand) provides standardized IP encapsulation over InfiniBand links. Therefore, all applications that require TCP/IP can continue to run without any modification. Detailed description of IPoIB can be found in [14]. InfiniBand also provides Socket Direct Protocol (SDP) [15] to accelerate the TCP/IP protocol via leveraging the RDMA capability, albeit its performance is still not as competive as the RDMA verbs. Similar to IPoIB, it requires no modification to the applications. Recently, OFED has announced to discontinue the support for SDP. However, we still include the evaluation results with SDP in this work to provide insight for its potential successor protocol, such as jVerbs. Although both InfiniBand and 10 GigE provide RDMA protocol, current Hadoop is unable to directly use it but via the SDP protocol.

In terms of the Hadoop setup, we employ the stable version Hadoop 1.0.4. During the experiments, one node is dedicated for both the NameNode of HDFS and the JobTracker of Hadoop MapReduce. On each slave nodes, we allocate 4 MapTask and 2 ReduceTask slots. The HDFS block size is chosen as 256MB as suggested by [16] to balance the parallelism and performance of MapTasks. We assign 512 MB and 1 GB heap size to each MapTask and ReduceTask respectively.

Benchmarks we adopt to conduct the evaluation include Terasort, WordCount, from standard Hadoop package. Terasort is extensively used through the entire evaluation due to its popularity as a *de facto* standard Hadoop I/O benchmark. In the Terasort, the size of intermediate data and the final output are as large as the input size. Via controlling the input data size, Terasort can effectively expose the I/O bottleneck across the Hadoop data processing pipeline.

Many cases have been explored in our evaluation experiments. To avoid confusion, we list the protocol and network environment used for each test case in Table 1. In addition, in the following sections, we use Hadoop-A and Hadoop Acceleration interchangeably.

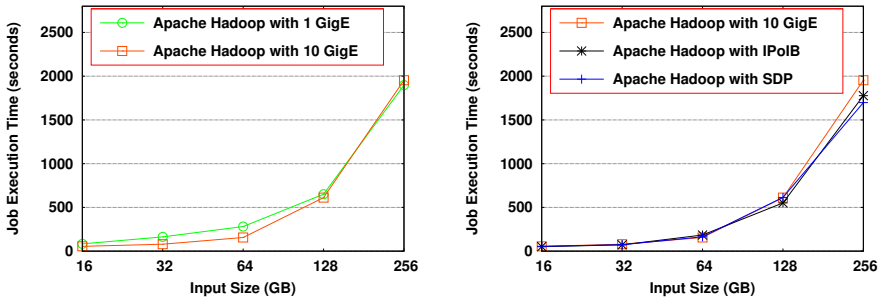
Table 1: Protocol and Network Description of Test Cases

Name of Test Cases	**Transport Protocol**	**Network**
Apache Hadoop with 1GigE	TCP/IP	1 GigE
Apache Hadoop with 10GigE	TCP/IP	10 GigE
Apache Hadoop with IPoIB	IPoIB	InfiniBand
Apache Hadoop with SDP	SDP	InfiniBand
Hadoop-A with 10 GigE	TCP/IP	10 GigE
Hadoop-A with RoCE	RoCE	10 GigE
Hadoop-A with RDMA	RDMA	InfiniBand
Hadoop-A with IPoIB	IPoIB	InfiniBand

3.2 Impact of High-Performance Interconnects on Hadoop MapReduce

We have evaluated the impact of high-performance interconnects on Apache Hadoop MapReduce from three aspects, which are *scalability*, *performance impact on different phases*, and *resource utilization*.

Scalability We study the scalability of Apache Hadoop via examining its ability to process a growing amount of data with fixed amount of computational resources and its ability to improve the throughput when expending the resources (a.k.a *Scale Up*).



(a) Comparison between 1 GigE and 10 GigE (b) Comparison between 10 GigE and InfiniBand

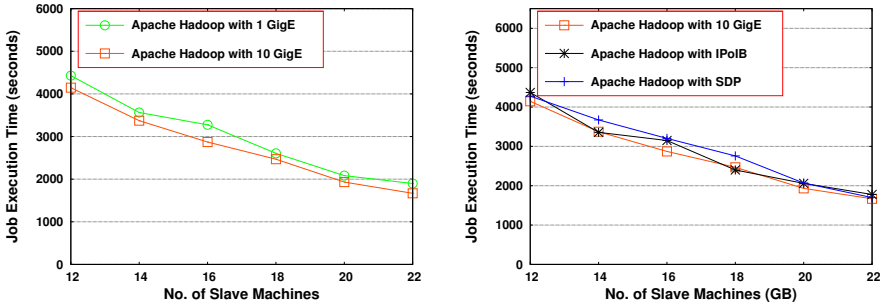
Fig. 3: Performance of Apache Hadoop with Growing Data Sizes

To investigate the ability of Hadoop to process growing amount of data, we run Terasort jobs of different input sizes on 22 slave nodes in both InfiniBand and Ethernet

environments. For each data size, we conduct 3 experiments and report the average job execution time. Overall, Hadoop shows linear scalability for small data sizes and nonlinear increase for large data sets.

Figure 3 shows the performance of Apache Hadoop with various data sizes under 3 different networks. As shown in Figure 3 (a), compared to running Hadoop on 1 GigE, using 10 GigE reduces the job execution time by 26.5% on average. Noticeably, fast network is very beneficial for small data sizes ($\leq$ 64 GB). For instance, when the data size is 32GB, compared to Hadoop on 1GigE, using 10 GigE effectively speeds up the job execution time by as much as 51.5%. This is because the movement of small size data is less dependent on disks and most of them reside in disk cache or system buffers. Thus high-performance networks can exhibit better benefits for data shuffling. While, simply adopting fast networks provide no encouraging improvements for large data sets ($\geq$ 128 GB) due to severe disk I/O bottleneck caused by large data sets. Such disk bottleneck is triggered in many places across the data shuffling. In particular, when the merge process is heavy, a large number of small random reads for retrieving the merged results exist in ReduceTasks and quickly vanish the improvements gained from fast data movement.

On the other hand, as shown in the Figure 3 (b), although InfiniBand provides even higher throughput and lower latency than 10 GigE, using InfiniBand achieves negligible performance improvements across the tests and only slightly reduces the job execution time by 13% for 256 GB data size due to less memory copy overhead involved in IPoIB and SDP.

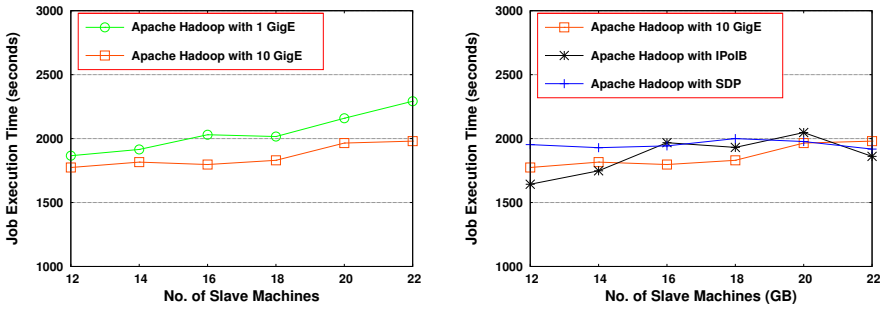


(a) Comparison between 1 GigE and 10 GigE (b) Comparison between 10 GigE and InfiniBand

Fig. 4: Strong Scaling Evaluation of Apache Hadoop

We further study the Apache Hadoop's ability to scale up with two patterns, which are *Strong Scaling* pattern and *Weak Scaling* pattern. In the case of strong scaling, we fix the input data size (256GB) while expending the number of slave nodes. In the case of weak scaling, for each test case, we use a fixed-size data size (6GB) for each ReduceTask, so the total input size increases linearly when we increase the number of nodes, reaching 264GB when 22 slave nodes are used.

Figure 4 shows the results of strong scaling tests under different networks. In all of the test cases, job execution time reduces linearly as more nodes join the computation.



(a) Comparison between 1 GigE and 10 GigE (b) Comparison between 10 GigE and InfiniBand

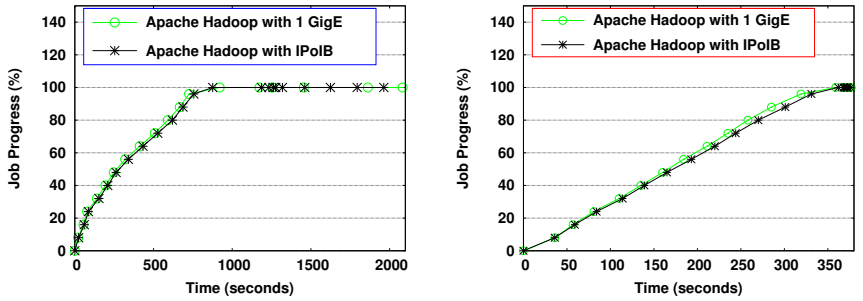
Fig. 5: Weak Scaling Evaluation of Apache Hadoop

However, as shown in the Figure 4 (a), on average, 10 GigE only marginally accelerates the job execution by 8.1% due to disk constraints. In addition, we observe that in the strong scaling case, 10 GigE even delivers better performance than InfiniBand. On average, 10 GigE outperforms IPoIB and SDP by 3.2% and 5.8%. Such results demonstrate that InfiniBand do not have superior advantages over 10 GigE for data-intensive MapReduce applications.

For the weak scaling tests, the optimal result should be a uniform execution time across the tests. However, as shown in Figure 5, in the 1 GigE environment, job is slowed down by 22.8% when the number of nodes increases from 12 to 22, showing poor scalability of 1 GigE. In contrast, using 10 GigE not only decreases the job execution time by 21.9% on average, but achieves better scalability (11.6% increase from 12 to 22 nodes) as well. Similar to the strong scaling tests, 10 GigE outperforms InfiniBand in the weak scaling test with respect to the job execution time. This is because when the number of nodes is small, 10 GigEs TCP/IP protocol is more lightweight than IPoIB protocol, which is an emulation of TCP/IP protocol in the InfiniBand environment, leading to more overhead. But InfiniBand shows better scalability for large cluster size due to higher bandwidth and the design of InfiniBand HCA. When the number of nodes increases from 12 to 22, job execution is slightly degraded by 1.7% when running with SDP protocol.

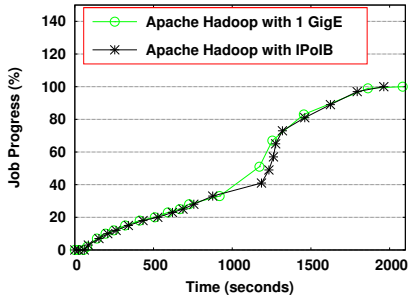
Impact on Different Phases of the Data Processing Pipeline To evaluate the impact of high-performance interconnects on different map and reduce phases of different types of Apache Hadoop applications. We employ two applications, Terasort and WordCount, which represent data-intensive and computation-intensive application, respectively. Figure 6 depicts the progress of map and reduce phases of different jobs.

Figure 6 (a) and (b) show that simply running Apache Hadoop on InfiniBand gain imperceptible performance improvement in both applications. In Terasort job, each MapTask spends a large portion of task execution time on merging the temporary spilled files. While the MapTasks in WordCount consume more CPU resources. Thus both types of MapTasks provide InfiniBand with limited optimization spaces. While on the ReduceTask side, as shown in Figure 6 (c) and (d), we observe that InfiniBand still fails to accelerate the progress of ReduceTasks of Terasort due to extremely slow merge

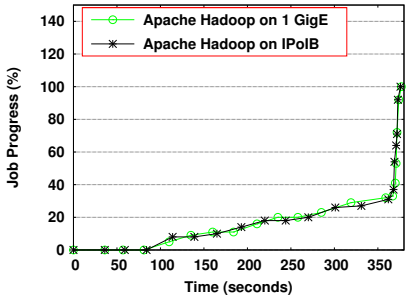


(a) Progress of Map Phase (Terasort)

(b) Progress of Map Phase (WordCount)



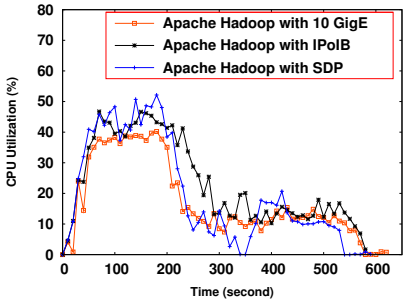
(c) Progress of Reduce Phase (Terasort)



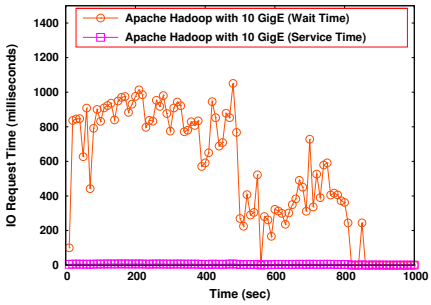
(d) Progress of Reduce Phase (WordCount)

Fig. 6: Impact of InfiniBand on Map and Reduce Phases

process within the ReduceTasks. Expensive merge process and repetitive merge behavior significantly drag down the draining of data from the network links, resulting in slow ReduceTasks. For the WordCount, InfiniBand offers negligible improvement due to very small intermediate data sizes. Since 10 GigE exibits similar performance pattern, we omit to elaborate on its results here for conciseness.



(a) CPU Utilization



(b) Disk Utilization

Fig. 7: Analysis of CPU and Disk Utilization of Apache Hadoop

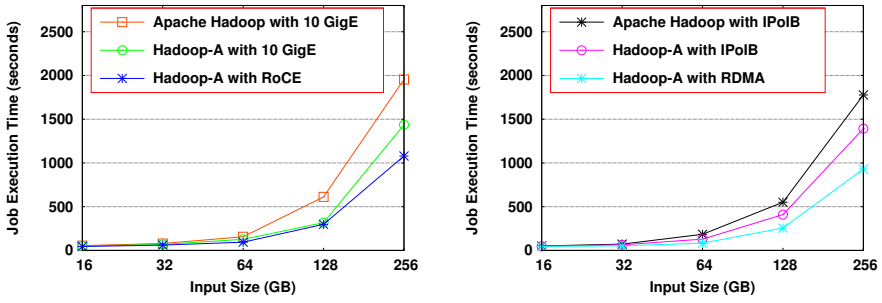
Resource Utilization In addition to job execution times, we have also collected the CPU and disk utilization statistics across the experiments. CPU utilization is an important performance metric. Low CPU utilization during data shuffling and merging can spare more CPU cycles for acceleration computation phases of Hadoop applica-

tions. Figure 7 (a) shows the CPU utilization of Apache Hadoop when running Terasort benchmark with 128GB input data. As shown in this figure, utilization difference between 10 GigE and InfiniBand is not significant. During the first 200 seconds, 10 GigE presents slightly lower utilization. The reason is that in the InfiniBand tests cases, there are more remote MapTasks, meaning more MapTasks need to fetch input data remotely. After 200 seconds, three networks achieve very similar utilization statistics. Compared to the 1 GigE, three networks reduces the CPU utilization by 9.6% on average (for clear presentation, we omit the 1 GigE data in Figure 7 (a)).

As we have claimed before, disk bound merge process is the major performance bottleneck that prevents Apache Hadoop from making full use of advantages of fast networks. To illustrate such issue, we have examined the I/O wait (queuing) time and the service time of I/O requests during a 160 Terasort job execution. As shown in Figure 7 (b), the I/O wait time can be more than 1000 milliseconds. Worse yet, extremely quick service time ($\leq$ 8 milliseconds) indicates that most I/O requests are spending nearly 100% of their time waiting in the queue , which explicitly demonstrates that the storage system is not able to keep up with the requests. In addition, both MapTasks and ReduceTasks intensively competes for disk bandwidth, this can significantly overload the disk subsystem, causing high-performance networks unable to accelerate the data-intensive MapReduce applications.

3.3 Impact of High-Performance Interconnects on Hadoop Acceleration

Hadoop Acceleration framework is a major step forward for Hadoop to support high-performance networks. Its network layer is designed to be highly portable on a variety of networks, such as InfiniBand and 10 GigE with a rich set of protocol support, such as RDMA, TCP/IP, etc. In this section, we study the performance of Hadoop Acceleration framework (Hadoop-A) on different networks via the same strategies used in section 3.2 and compare its performance against original Apache Hadoop.



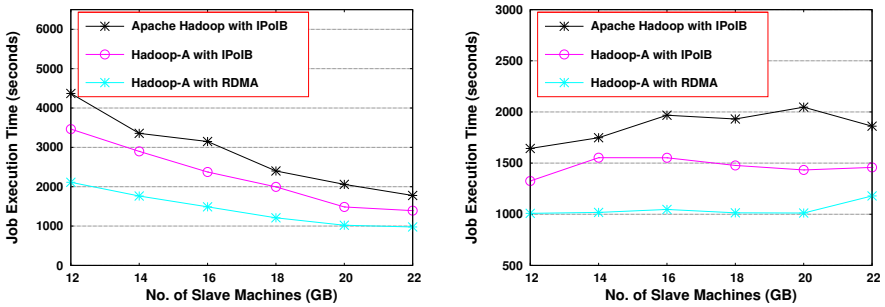
(a) Performance of Hadoop-A in 10 GigE Environment

(b) Performance of Hadoop-A in InfiniBand Environment

Fig. 8: Performance of Hadoop Acceleration with Growing Data Sizes

Scalability Figure 8 shows the performance of Hadoop-A with growing data sets by conducting Terasort benchmark. Overall, Hadoop-A is superior to the original Apache

Hadoop for data-intensive applications. In the 10 GigE environment, compared to Apache Hadoop, running Hadoop-A with TCP/IP reduces the job execution times by 19.3% on average. Enabling the RDMA over Converged Ethernet (RoCE) protocol further accelerates the job execution by up to 15.3%. In the InfiniBand environment, on average, Hadoop-A outperforms the original Hadoop by 14.1% and 38.7%, when IPoIB and RDMA protocols are used respectively. Moreover, in both environments, we observe that Hadoop-A delivers better performance for all ranges of data sizes. For small data sets, Hadoop-A is better due to its elimination of JVM overhead from the critical path of data shuffling. For large data sets, Hadoop-A mitigates the disk bottleneck via its flexible framework to support various merging algorithms, such as Network Levitated Merge [10] and Hierarchical Merge algorithm [12], both of which fully take advantage of the memory to avoid disk I/O for the intermediate data merging process at the ReduceTask side.

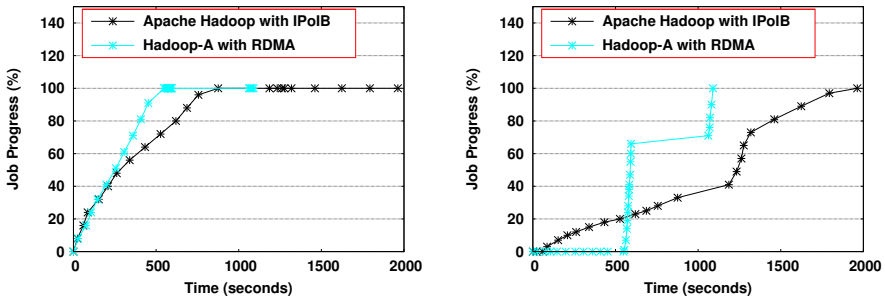


(a) Strong Scaling of Hadoop Acceleration (b) Weak Scaling of Hadoop Acceleration

Fig. 9: Scalability Evaluation of Hadoop Acceleration in InfiniBand Environment

Scalability is a critical feature for MapReduce, thus we also evaluate the strong scaling and weak scaling capability of Hadoop-A. As shown in the Figure 9 (a), running Hadoop-A with RDMA and with IPoIB effectively outperform Apache Hadoop by 49.5% and 20.9%, respectively on average, and achieves a comparable linear reduction as the original Hadoop. On the other hand, for the weak scaling tests, Hadoop-A with RDMA and with IPoIB reduce the execution time by 43.6% and 21.1%, respectively on average, compared to Apache Hadoop with IPoIB, and exhibit slight performance variance across the tests. (Similar performance is observed under the 10 GigE environment, so we omit the results for succinctness).

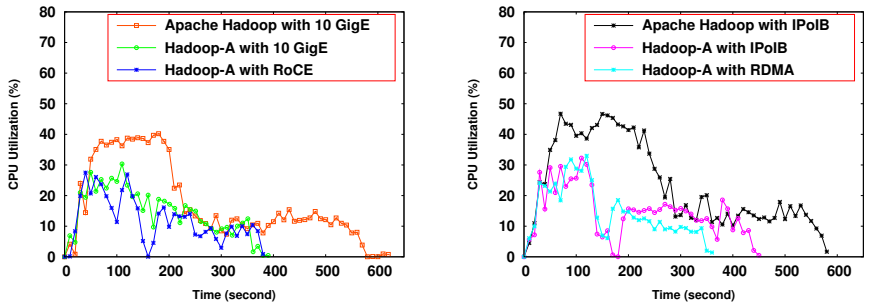
Impact on Different Phases of Data Processing Pipeline Across the tests, we observe that Hadoop-A is able to speedup both the map and reduce phases of the data-intensive applications. Figure 10 illustrates such phenomenon, in which MapTasks of TeraSort complete much faster under Hadoop-A, especially when the percentage of completion goes over 50%. This is because in Hadoop-A, all ReduceTasks only performs lightweight operations such as fetching headers and setting up in-memory priority queue during the map phase, thereby leaving more resources such as disk bandwidth for MapTasks, which can fully relish the resources to accelerate their executions. While, on



(a) Map Progress (b) Reduce Progress

Fig. 10: Impact on Different Phases of Terasort

the ReduceTask side, we observe slow progress at the beginning (before 500th second). This is because during this period, ReduceTasks are just constructing the in-memory priority queue, meanwhile waiting for the map phase to complete. As soon as the map phase complete (after 500th second), ReduceTask rapidly progresses to the 60%, significantly outperforming the original ReduceTasks.



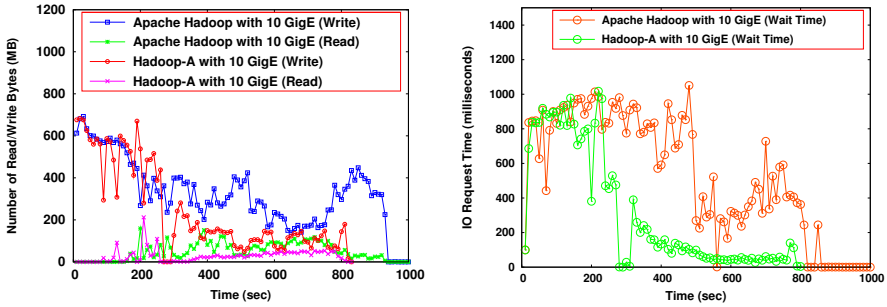
(a) 10 GigE Environment (b) InfiniBand Environment

Fig. 11: CPU Utilization of Hadoop Acceleration

Resource Utilization By eliminating the overhead of JVM and reducing the disk I/O, Hadoop-A greatly lowers the CPU utilization. As shown in the Figure 11 (a), in the 10 GigE environment, compared to Apache Hadoop on 10GigE, Hadoop-A on RoCE and on 10GigE reduce the CPU utilization by 46.4% and 33.9%, respectively on average. In addition, compared to TCP/IP protocol, leveraging RoCE cuts down on the CPU utilization by about 18.7% due to less memory copies to consume the CPU cycles. Such improvement is also observed in the InfiniBand environment as shown in the Figure 11 (b).

Table 2: I/O Blocks

	READ (MB)	WRITE (MB)
Apache Hadoop with 10 GigE	5,426	36,427
Hadoop-A with 10 GigE	2,441	22,713



(a) I/O Profiling (b) I/O Wait Time

Fig. 12: Disk Utilization of Hadoop Acceleration

To assess the effectiveness of Hadoop-A to improve the disk utilization, we have also measured the disk accesses during data shuffling under Hadoop-A, and compared the results with that of Apache Hadoop. We run TeraSort on 20 slave nodes with 160GB as input size. On each node we run vmstat and iostat to collect I/O statistics and trace the output every 2 seconds.

Table 2 shows the comparison of the number of bytes read and written by Hadoop and Hadoop-A from and into local disks per slave node. Overall, Hadoop-A reduces the number of read blocks by 55.1% and write blocks by 37.6%. This demonstrates that Hadoop-A reduces the number of I/O operations and relieves the load of underlying disks.

Fig. 12 (a) shows the progressive profile of read and write bytes during the job execution. During the first 200 seconds in which MapTasks are active, there is no substantial difference between Hadoop and Hadoop-A in terms of disk I/O traffic. After the first 200 seconds, ReduceTasks start fetching and merging the intermediate data actively. Because Hadoop-A uses the network-levitated merge algorithm which completely eliminates the disk access for the shuffling and merging of data segments, we observe that Hadoop-A effectively reduces the number of bytes read from or written to the disks. Therefore, disk I/O traffic is significantly reduced during this period.

As shown in section 3.2, I/O requests in Apache Hadoop experience long I/O wait time, thus degrading the performance. In order to further analyze the benefit from the reduced disk accesses, we measure the I/O wait time in Hadoop-A, and compare it with Apache Hadoop. The result is shown in Figure 12 (b). As shown in the figure, Hadoop-A leads to similar or lower I/O wait time during the first 200 seconds, which corresponds to the mapping phase of the job execution. As the job progresses, I/O wait time of Hadoop-A is significantly reduced when job enters into the shuffle/merge and reduce phases. This demonstrates that the reduction of disk accesses contributes to the reduction of I/O wait time. Aggregately, these experiments indicate that Hadoop-A effectively improves I/O performance in Hadoop, thereby effectively shortening job execution time.

4 Related Work

Leveraging high performance interconnects to move data in the Hadoop ecosystem has attracted numerous research interests from many organizations over the past a few years. Huang *et al.* [17] designed an RDMA-based HBase over InfiniBand. In addition, they pointed out the disadvantages of using Java Socket Interfaces in Hadoop ecosystem. A recent evaluation [18] of Hadoop Distributed File system (HDFS) [11] used the SDP [15] and IPoIB protocols of InfiniBand [19] to investigate the potential benefit of leveraging fast networks for pipelined writing in HDFS. In the same work, authors showed that Hadoop was unable to directly leverage the RDMA (Remote Direct Memory Access) communication mechanism available from high-performance RDMA interconnects. For that reason, to enhance the efficiency of HDFS, Islam *et al.* [20] modified HDFS network connection structure to use RDMA over InfiniBand via JNI interfaces. Jose *et al.* [21, 22] implemented a scalable memcached through taking advantage of performance benefits provided by InfiniBand. But, although Hadoop MapReduce is a fundamental basis of Hadoop ecosystem, there is lack of research on how to efficiently leverage high performance interconnects in Hadoop MapReduce. [10] studied the feasibility of importing RDMA support into the Apache Hadoop. Meanwhile, [23] measured the overhead imposed by Java Virtual Machine on the Hadoop shuffling.

Adopting RDMA from high speed networks for fast data movement has been very popular in various programming models and storage paradigms, such as MPI. [24] studied the pros and cons of using RDMA capabilities in a great details. Liu *et al.* [25] designed RDMA-based MPI over InfiniBand. Yu *et al.* [26] implemented a scalable connection management strategy for high-performance interconnects. Implementations of PVFS [27] on top of RDMA networks such as InfiniBand and Quadrics were described in [28] and [29], respectively. However, none of them have studied the impact of high-speed inter-connection on Hadoop MapReduce framework.

5 Conclusions

In the Hadoop MapReduce framework, data shuffling accounts for a considerable portion of the total execution time of MapReduce programs. Meanwhile, the current technologies in interconnect fabric have made the speed of NIC comparable with RAM-base memory. In this paper, we undertake a comprehensive evaluation on how Hadoop MapReduce framework can be accelerated by leveraging high-performance interconnects. We have examined the performance of Apache Hadoop and Hadoop Acceleration framework on InfiniBand and 1/10 Gigabit Ethernet from the aspects of scalability, data processing pipeline and resource utilization. Our experiment results reveal that simply switching to the high-performance interconnects cannot effectively boost the performance of Apache Hadoop due to the cost imposed by JVM and disk bottleneck on Hadoop intermediate data shuffling. Moreover, with various application evaluated on both Ethernet and InfiniBand environments, we demonstrate that Hadoop Acceleration framework can significantly reduce the CPU utilization and job execution time for MapReduce jobs that generate a large amount of intermediate data. Specifically, Hadoop Acceleration can effectively reduce the execution time of Hadoop jobs by up to

49.5% and lower the CPU utilization by 46.4%. In the future, we plan to further evaluate the MapReduce programs on large clusters that consists of hundreds of thousands nodes.

References

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

[2] Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. In: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6. OSDI'04, Berkeley, CA, USA, USENIX Association (2004) 10–10

[3] Pavlo, A., Paulson, E., Rasin, A., Abadi, D.J., DeWitt, D.J., Madden, S., Stonebraker, M.: A comparison of approaches to large-scale data analysis. In: Proceedings of the 35th SIGMOD international conference on Management of data. SIGMOD '09, New York, NY, USA, ACM (2009) 165–178

[4] Condie, T., Conway, N., Alvaro, P., Hellerstein, J.M., Elmeleegy, K., Sears, R.: Mapreduce online. In: Proceedings of the 7th USENIX conference on Networked systems design and implementation. NSDI'10, Berkeley, CA, USA, USENIX Association (2010) 21–21

[5] Seo, S., Jang, I., Woo, K., Kim, I., Kim, J.S., Maeng, S.: HPMR: Prefetching and preshuffling in shared MapReduce computation environment. In: CLUSTER. (August 2009) 1–8

[6] Rao, S., Ramakrishnan, R., Silberstein, A., Ovsiannikov, M., Reeves, D.: Sailfish: a framework for large scale data processing. In: Proceedings of the Third ACM Symposium on Cloud Computing. SoCC '12, New York, NY, USA, ACM (2012) 4:1–4:14

[7] InfiniBand Trade Association: The InfiniBand Architecture. http://www.infinibandta.org

[8] Recio, R., Culley, P., Garcia, D., Hilland, J.: An rdma protocol specification (version 1.0) (October 2002)

[9] : High Performance Computing (HPC) on AWS. http://aws.amazon.com/hpc-applications/

[10] Wang, Y., Que, X., Yu, W., Goldenberg, D., Sehgal, D.: Hadoop acceleration through network levitated merge. In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. SC '11, New York, NY, USA, ACM (2011) 57:1–57:10

[11] Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). MSST '10, Washington, DC, USA, IEEE Computer Society (2010) 1–10

[12] Que, X., Wang, Y., Xu, C., Yu, W.: Hierarchical merge for scalable mapreduce. In: Proceedings of the 2012 workshop on Management of big data systems. MBDS '12, New York, NY, USA, ACM (2012) 1–6

[13] : Open Fabrics Alliance. http://www.openfabrics.org.

[14] J. Chu and V. Kashyap: Transmission of IP over InfiniBand(IPoIB). http://tools.ietf.org/html/rfc4391 (2006)

[15] InfiniBand Trade Association: Socket Direct Protocol Specification V1.0 (2002)

[16] Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., Stoica, I.: Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In: Proceedings of the 5th European conference on Computer systems. EuroSys'10, New York, NY, USA, ACM (2010) 265–278

[17] Huang, J., Ouyang, X., Jose, J., ur Rahman, M.W., Wang, H., Luo, M., Subramoni, H., Murthy, C., Panda, D.K.: High-performance design of hbase with rdma over infiniband. In: 26th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2012, Shanghai, China, May 21-25, 2012. IPDPS'12 (2012) 774–785

[18] Sur, S., Wang, H., Huang, J., Ouyang, X., Panda, D.K.: Can High-Performance Interconnects Benefit Hadoop Distributed File System? In: MASVDC-2010 Workshop in conjunction with MICRO. (Dec 2010)
[19] Infiniband Trade Association. http://www.infinibandta.org
[20] 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: Proceedings of 2012 International Conference for High Performance Computing, Networking, Storage and Analysis. SC'12, ACM (2012)
[21] Jose, J., Subramoni, H., Luo, M., Zhang, M., Huang, J., ur Rahman, M.W., Islam, N.S., Ouyang, X., Wang, H., Sur, S., Panda, D.K.: Memcached design on high performance rdma capable interconnects. In: ICPP, IEEE (2011) 743–752
[22] Jose, J., Subramoni, H., Kandalla, K., Wasi-ur Rahman, M., Wang, H., Narravula, S., Panda, D.K.: Scalable memcached design for infiniband clusters using hybrid transports. In: Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012). CCGRID'12, Washington, DC, USA, IEEE Computer Society (2012) 236–243
[23] Wang, Y., Xu, C., Li, X., Yu, W.: Jvm-bypass for efficient hadoop shuffling. In: 27th IEEE International Parallel and Distributed Processing Symposium. IPDPS'13, IEEE (2013)
[24] Frey, P.W., Alonso, G.: Minimizing the hidden cost of rdma. In: Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems. ICDCS'09, Washington, DC, USA, IEEE Computer Society (2009) 553–560
[25] Liu, J., Wu, J., Panda, D.K.: High performance rdma-based mpi implementation over infiniband. International Journal of Parallel Programming **32** (2004) 167–198
[26] Yu, W., Gao, Q., Panda, D.K.: Adaptive connection management for scalable mpi over infiniband. In: Proceedings of the 20th international conference on Parallel and distributed processing. IPDPS'06, Washington, DC, USA, IEEE Computer Society (2006) 102–102
[27] P. H. Carns and W. B. Ligon III and R. B. Ross and R. Thakur: PVFS: A Parallel File System For Linux Clusters. In: Proceedings of the 4th Annual Linux Showcase and Conference, Atlanta, GA (October 2000) 317–327
[28] Wu, J., Wychoff, P., Panda, D.K.: PVFS over InfiniBand: Design and Performance Evaluation. In: Proceedings of the International Conference on Parallel Processing '03, Kaohsiung, Taiwan (October 2003)
[29] Yu, W., Liang, S., Panda, D.K.: High Performance Support of Parallel Virtual File System (PVFS2) over Quadrics. In: Proceedings of The 19th ACM International Conference on Supercomputing, Boston, Massachusetts (June 2005)