Performance Characterization and Acceleration of Big Data Workloads on OpenPOWER System

Xiaoyi Lu, Haiyang Shi, Dipti Shankar, and Dhabaleswar K. (DK) Panda
Department of Computer Science and Engineering, The Ohio State University
{lu.932, shi.876, shankar.50, panda.2}@osu.edu

***Abstract*—IBM's POWER processor has been advocated as the high-performance architecture designed for processing Big Data workloads. With the collaborations through the OpenPOWER Foundation, more and more innovations for POWER architecture are emerging to solve Big Data challenges. For example, with the cooperation between IBM and Mellanox, the latest generation of Remote Direct Memory Access (RDMA) capable InfiniBand network can deliver tremendous performance on POWER processors. On the other hand, many RDMA-based designs and optimizations recently have been proposed in the community for accelerating big data processing systems (such as Apache Hadoop and Spark). However, these studies mostly focus on achieving higher performance over Intel Xeon or other x86 architectures. As OpenPOWER systems are getting momentum, we set out to answer the question how much can the RDMA-based communication runtime benefit Big Data processing middleware running over OpenPOWER systems as compared to the default TCP/IP-based designs. To answer this question, this paper first presents an extensive performance characterization on RDMA-based Hadoop RPC engine over OpenPOWER system. We further propose new designs to enable efficient CPU affinity policies and architecture-aware tuning in the RDMA-based communication engine for Hadoop and Spark. With these various accelerations, our performance evaluation shows that our proposed designs can achieve up to 2.73X performance improvement for Hadoop RPC benchmark as compared to default Hadoop running with IP-over-IB protocol on OpenPOWER systems. In addition, our proposed design can gain up to 29.37% performance improvement for Hadoop and Spark workloads as compared to the default RDMA designs running on an OpenPOWER cluster.**

I. Introduction

In 2013, IBM and other companies launched the OpenPOWER Foundation [13], which is an open technical membership organization that aims to customize POWER CPU processors and system platforms for optimization and innovation in advanced hardware technologies and software co-designs. IBM POWER processor (e.g., POWER8) has been advocated as the high-performance architecture designed for processing Big Data workloads. The OpenPOWER Foundation has established that they will focus on the following four technical areas: (1) Machine Learning/AI, (2) Database/Analytics, (3) Cloud, and, (4) Containers; as a part of their 2017 goals. To deliver high-performance for these various workloads, OpenPOWER Foundation members have worked together closely to push more and more innovations to OpenPOWER-enabled systems. For example, with the collaboration between IBM and Mellanox, the latest generation of Remote Direct Memory Access (RDMA) capable InfiniBand network is being used to deliver high-performance communication on OpenPOWER clusters. With all of these developments, OpenPOWER systems are quickly becoming the choice of processing power for High-End Computing and High-Performance Computing (HPC) clusters.

With the increase in the amount of valuable data being generated at various businesses and organizations, Big Data combined with HPC is creating new solutions area of biomedical research, business analytics and security, smart energy grids, manufacturing, and many more. With the convergence of HPC and Big Data Analytics, a new type of workload called High-Performance Data Analytics (HPDA) is becoming the next game-changing business opportunity. According to IDC's report, HPDA use cases are rapidly emerging in the HPC market which is expected to reach $31 billion by 2019 [18]. In order to handle HPDA workloads efficiently, many novel accelerations and designs have been proposed to improve the performance of current-generation Big Data middleware. Remote Direct Memory Access (RDMA) based communication stacks have become a research hotspot in the field of High-Performance Big Data analytics, for accelerating Big Data processing systems such as Hadoop [28, 33, 44], Spark [34, 35], HBase [25, 36], and Key-Value Stores [23, 30, 46].

A. Motivation

Table I summarizes recently state-of-the-art research in literature pertaining to Big Data middleware for HPC clusters. We broadly categorize them based on the following three aspects: (1) architecture, (2) network, and, (3) whether there are native RDMA-based designs proposed for these environments. As we can see, most of these studies are focused on achieving high-performance on Intel Xeon or other x86 architectures with RDMA-capable networks (such as InfiniBand or RoCE) or high-speed Ethernet. Additionally, some recent studies [1, 12, 15, 29, 42] have been presented on running Big Data workloads over OpenPOWER systems with InfiniBand EDR or 10/40/100 GigE networks. However, these designs are typically using TCP/IP, IP-over-InfiniBand (IPoIB), or RSockets [24] protocols, which are not based on native RDMA designs. Table I clearly shows that the Big Data community still needs more research effort on POWER architecture to study the performance characteristics and propose more accelerations for native RDMA-based Big Data processing middleware.

As OpenPOWER systems are getting momentum, we set out to answer the question: **How much can the RDMA-based communication runtime benefit Big Data processing middleware running over OpenPOWER architecture as compared to the default TCP/IP-based designs?** To answer this question, it is therefore vital to address the following challenges:

- What are the performance characteristics of representative RDMA-based Big Data processing middle-

This research is supported in part by National Science Foundation grants #IIS-1447804, #ACI-1664137, #IIS-1636846, and #CNS-1513120.

ware running over OpenPOWER systems with modern POWER processors and InfiniBand networks?

- Can RDMA-based Big Data processing middleware that were originally designed for the x86 architecture demonstrate similar performance benefits on OpenPOWER systems, as compared to other communication protocols?
- Based on these studies, can we propose new accelerations for Big Data processing middleware on OpenPOWER systems to attain even better benefits as compared to the default RDMA-based designs?
- How much can overall performance benefits be gained in typical Big Data application workloads on OpenPOWER systems by leveraging these optimized RDMA-based designs?

TABLE I: Comparison with Related Work

Work	Architecture	Network	Native RDMA based Design
[6, 28, 34, 35, 44, 46, 49]	x86	InfiniBand/RoCE	Yes
[5, 41]	x86	10/40/100 GigE	No
[1, 42]	POWER	InfiniBand	No
[12, 15, 29]	POWER	10/40/100 GigE	No
This paper	POWER	InfiniBand	Yes

B. Contribution

To address all of these challenges, this paper first presents an extensive performance characterization of RDMA-based Hadoop RPC engine over an OpenPOWER system. Since Hadoop RPC is the most fundamental communication mechanism in Hadoop ecosystem and represents communications conforming to a wide range of distributed operations in Big Data middleware, we believe that Hadoop RPC is a good starting point of investigating the performance characteristics of RDMA-based communication engine on OpenPOWER systems. Through these performance characterizations, we find that the RDMA-based communication engine can deliver the similar higher performance benefits observed on x86 clusters as compared to the default design running over the IPoIB protocol, on OpenPOWER systems. In the meantime, we observe the performance variations for both RDMA and IPoIB on OpenPOWER platforms. However, the same designs running over Intel Xeon x86 platforms are showing more stable performance numbers. To further investigate the reasons behind these variations in performance, we analyze how CPU affinity and JDK versions impact performance on OpenPOWER systems.

Based on these studies and in-depth analysis, we propose new designs to enable efficient CPU affinity policies and architecture-aware tuning in RDMA-based communication engines for Hadoop and Spark like Big Data processing frameworks. With these various accelerations, our proposed designs can achieve up to 2.73X performance improvement for Hadoop RPC benchmark as compared to default Hadoop running with IP-over-IB protocol and up to 29.37% performance improvement for Hadoop and Spark workloads as compared to the default RDMA designs running on an OpenPOWER cluster. In addition, our proposed designs enable us to attain stable performance gains by employing OpenPOWER clusters for Big Data processing workloads. To the best of our knowledge, this is the first paper to discuss how to achieve higher and stable performance on OpenPOWER systems with native RDMA-based designs for popular Big Data processing middleware such as Hadoop and Spark.

The rest of the paper is organized as follows. Section II presents the background of POWER architecture, InfiniBand, and RDMA-based designs for Apache Hadoop and Spark. We further discuss the performance characterizations in Section III. Section IV presents our proposed design. Section V describes our detailed evaluation. Section VI discusses related work. We conclude in Section VII with future work.

II. Background

In this section, we present a brief background on POWER architecture, InfiniBand, and RDMA-based designs for Apache Hadoop and Spark.

A. Overview of POWER Architecture

IBM's POWER architecture [31] is considered the cornerstone of innovation of the OpenPOWER Foundation [13], that promotes the adoption of an open server architecture for data centers and compute clusters across various organizations around the world. POWER8 [14], represented in Figure 1, is the first processor supporting OpenPOWER. Each POWER8 CPU supports up to 12 cores per socket, with each core supporting 8 hardware Simultaneous Multi-Threading (SMT) threads, 64 KB L1 data cache and 32 KB L1 instruction cache. In addition to this, each core supports a 512KB of L2 cache and 8MB of L3 cache, along with up to 128 MB off-chip L4 eDRAM. Further, the on-chip memory controllers can handle 1 TB of RAM and 230 GB/s sustained memory bandwidth and 48 GB/s of I/O to other parts of the system. The cores are designed to operate at clock rates between 2.5 and 5 GHz.

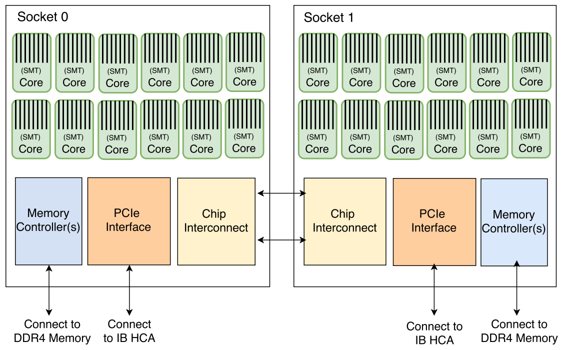


Fig. 1: Overview of IBM POWER8 architecture

POWER8 also has enhanced prefetching features such as instruction speculation awareness and data prefetch depth awareness. It also introduces CAPI (Coherence Attach Processor Interface) [3], which is a direct link into the CPU, allowing peripherals and co-processors (GPUs, FPGAs, etc.) to communicate directly with the POWER8 CPU. These POWER8 servers have been designed to be suitable for Big Data analytical workloads, by providing 4 times more threads per core, memory bandwidth, and cache than other platforms. POWER9 [48] is IBM's successor to POWER8 that offers scale-out and scale-up variations with either a 12-core SMT8 or a 24-core SMT4 model.

B. InfiniBand and RDMA

InfiniBand (IB) [26] is an open industry-standard specification for data flow between server I/O and inter-server communication that offers high-performance, low CPU overhead, and scalability; with speeds of 56 Gb/sec (IB FDR), 100 Gb/sec (IB EDR), 200 Gb/sec (IB HDR), etc. IB networks provide advanced features such as Remote Direct Memory Access (RDMA), that allows software to remotely read or update memory contents of another remote process without involving the remote CPU. These features can be leveraged to design low-latency communication protocols through the use of the low-level Infiniband Verbs (ibverbs) library. InfiniBand software stacks, such as OpenFabrics [39], provide a driver for implementing the IP layer over InfiniBand (IP-over-IB). Several IBM POWER8 clusters from the OpenPOWER foundation are being equipped with Mellanox ConnectX-4 EDR IB adapters and switch systems [17].

C. RDMA-based Designs for Apache Hadoop and Spark

A majority of the existing HPC clusters today are equipped with modern high-speed interconnects, such as InfiniBand and 10/40 Gigabit Ethernet with RoCE/iWARP support, which offers high-performance and advanced features such as RDMA. This has driven the designs of new communication protocols for accelerating Big Data middleware such as Apache Hadoop and Spark [40]. Instead of using IPoIB or RSockets to translate TCP/IP-based communication protocols, these designs employ a native IB Verbs-based RDMA-aware communication engine that replaces Java Sockets channel employed in the default Apache Hadoop and Spark frameworks. The RDMA-based communication protocols benefit the large-scale I/O involved during HDFS write (with replication) [27, 28], high-speed control message exchange in Hadoop RPC [33], and intermediate data shuffling in MapReduce [43], and Spark frameworks [34, 35]. Specifically, for Apache Spark, the proposed RDMA-based communication engine servers as a pluggable block transfer service which can support all three shuffle schemes (sort, hash, tungsten-sort) [35], along with on-demand RDMA connection establishment, non-blocking pipelined data transfers, and slab-based buffer management [34].

While the RDMA-enabled communication engine has shown tremendous performance improvements for x86 processor based InfiniBand clusters, in this paper, we study the potential and optimize these designs for upcoming 'OpenPOWER + InfiniBand' clusters.

III. Performance Characterization with RDMA-based Hadoop RPC on OpenPOWER System

In this section, we first use Hadoop RPC as a communication-intensive workload to characterize the performance of RDMA-based communication engine on an OpenPOWER system. In order to contrast, we also choose a contemporary Intel Xeon x86 platform to conduct the same experiments. Through these contrast tests, we are able to show more insights into the performance characteristics. Table II shows the hardware and software specification of the clusters used in this paper.

TABLE II: Specification of the Clusters used in this paper

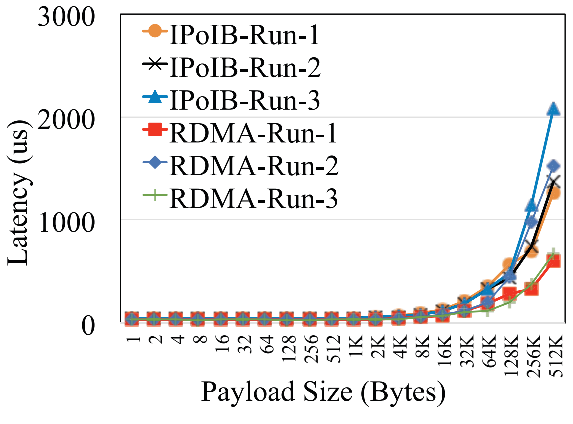
Specification	Xeon	OpenPOWER
Processor Family	Intel Broadwell	IBM POWER8
Processor Model	E5 v2680	PPC64LE
Frequency	2.4 GHZ	3.5 GHZ
No. of Sockets	2	2
Cores per Socket	14	10
Threads per Core	1	8 (SMT Threads)
Mesh Config	NUMA	NUMA
RAM (DDR)	512 GB	256 GB
Interconnect	IB-EDR (100 Gbps)	IB-EDR (100 Gbps)
OS	CentOS Linux release 7.2.1511	Red Hat Enterprise Linux Server release 7.2 (Maipo)
JDK	IBM JDK 1.8.0 OpenJDK 1.8.0	IBM JDK 1.8.0 OpenJDK 1.8.0

A. Performance Comparison of RDMA and IPoIB on OpenPOWER and Xeon x86 Platforms

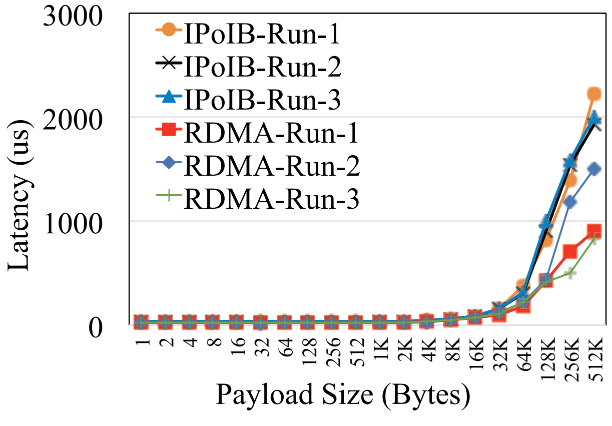
The first group of tests we performed compare the latency of RDMA-based Hadoop RPC (v2.7.3) and default Hadoop RPC running over IPoIB on OpenPOWER and Xeon x86 platforms. Figure 2 presents these results. We run Hadoop RPC latency micro-benchmark [37] tests on two OpenPOWER nodes. One node runs a server process and the other runs a client process. In the Hadoop RPC latency benchmark, it runs over 20K times of iterations for each message size and the first half of iterations are considered as warming-up iterations. Thus, the effect of JVM classloading, instance initialization, etc. will be reduced to the minimal. The numbers are finally reported by averaging 10K times of iterations. We undertake these tests on other platforms or configurations for the latency comparisons in the rest of this paper.

In Figure 2(a), we show the Hadoop RPC latency numbers with RDMA and IPoIB on IBM POWER8 ppc64le architecture. In this figure, we run three times of tests for each protocol. Overall, we see that RDMA can outperform IPoIB by achieving up to 2.3X performance speedup across different message sizes. However, if we look at different runs, we see obvious performance variations. For example, the typical RDMA-based Hadoop RPC only has around 600 us latency for 512 KB message size, while for the 'RDMA-Run-2' case, we see the latency becomes around 1,530 us. Similarly, for IPoIB, the typical latency for 512 KB message size is around 1,300 us, but for the 'IPoIB-Run-3' case, we see the latency goes up to 2,100 us. Such type of variation pushes us to explore what could be the reasons behind these numbers.

To figure out the possible reasons, we first conduct the same kind of experiments on Intel Xeon x86 platform as the numbers shown in Figure 2(b). From these numbers, we see that RDMA-based Hadoop RPC protocol can achieve up to around 3X performance speedup compared to IPoIB-based protocol across different message sizes. By comparing the numbers between OpenPOWER and Xeon, we find that Hadoop RPC has better performance on OpenPOWER system even though they are using the same generation of InfiniBand EDR network. This is mainly due to the higher CPU frequency of POWER8 cores, which will benefit the serialization and deserialization processes of RPC calls. Regarding variations, similar trends are also happening. These results make us even more doubting since typically we do not see such large variations for Xeon x86 systems during our regular tests.

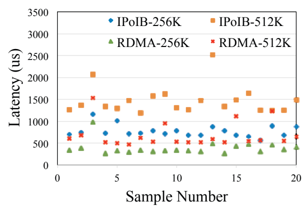


(a) IBM POWER8 ppc64le

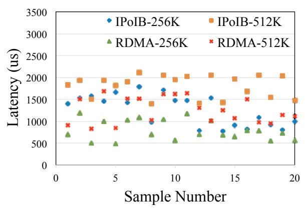


(b) Intel Xeon x86

Fig. 2: Performance Comparison of RDMA and IPoIB on POWER8 and Xeon x86 Platforms with IBM JDK 1.8

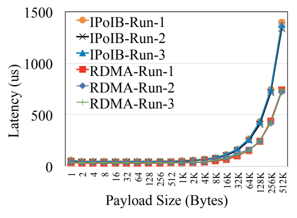


(a) IBM POWER8 ppc64le

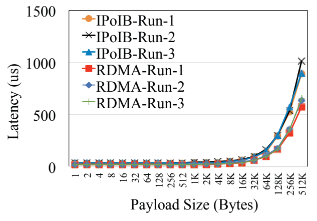


(b) Intel Xeon x86

Fig. 3: Variation Distributions of RDMA and IPoIB on POWER8 and Xeon x86 Platforms with IBM JDK 1.8



(a) IBM POWER8 ppc64le



(b) Intel Xeon x86

Fig. 4: Performance Comparison of RDMA and IPoIB on POWER8 and Xeon x86 Platforms with OpenJDK 1.8

To make sure the variations are not due to the use of sharing cluster resources, we run 20 times of tests across different times of three days. The numbers are drawn in Figure 3. The X-axis means the sample number of the test. The Y-axis means the latency of different runs. Interestingly, we do see variations on both platforms for both protocols. Even though RDMA still shows much better performance than IPoIB on both POWER8 and Xeon x86 platforms, we do want to reduce the variation as much as possible.

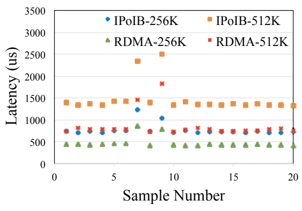
During these experiments, one factor comes into our mind, which is the JDK version being used in the above tests. Typically, for our regular tests on Xeon x86 systems, we use OpenJDK 1.7 or 1.8 in these days. However, since IBM JDK has been optimized for IBM OpenPOWER systems, we choose IBM JDK 1.8 as default JDK for performing above experiments. To confirm our regular stable numbers, we switch our JDK to OpenJDK 1.8 and rerun all of the above tests on both platforms. Figure 4 shows the performance comparison of RDMA and IPoIB on POWER8 and Xeon x86 platforms with OpenJDK 1.8. As we see from Figure 4, the three runs for both RDMA and IPoIB are much more stable than earlier numbers and RDMA can show clear benefit (around 2X performance speedup on POWER8 and x86) compared to the IPoIB protocol.

To confirm these results, we rerun the tests 20 times on both POWER8 and Xeon machines, and Figure 5 shows the the numbers. Compared to Figure 3, these numbers look much more stable, especially on Xeon x86 systems. This also matches the trend of our regular tests. However, on POWER8, there are still some outliers in the graph, of which the main reason is due to multiple SMT threads in each physical core. Note that on the Xeon x86 platform, the hyperthreading is disabled on CPUs and we see very stable numbers on Xeon. To disable SMT on POWER, we need the privileged permission, which is impossible to get on HPC clusters traditionally. That is why we skip the discussion on enabling and disabling SMT threads for OpenPOWER systems. Some research studies like [29] have discussed the SMT effect on the application performance.

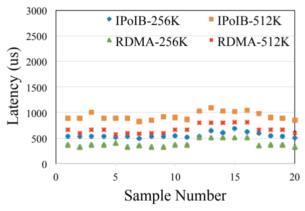
Since this paper focuses on how to achieve stable high performance on OpenPOWER machines, we want to stick to higher performance JDK on POWER8 machines. If we compare Figure 2(a) and Figure 4(a), even though IBM JDK shows larger variations compared to OpenJDK, it does show higher performance on OpenPOWER systems compared to OpenJDK. For example, for 512 KB message size, IBM JDK can give respectively around 600 us and 1,300 us for RDMA and IPoIB, while OpenJDK gives around 730 us and 1,400 us for RDMA and IPoIB typically. This means if we can reduce the variations for IBM JDK numbers, IBM JDK could become a good choice for gaining better performance on OpenPOWER systems.

B. Performance Impact of CPU Affinity for RDMA and IPoIB on OpenPOWER System

In the HPC field, whenever we see variations in parallel programs, one possible reason we should check is CPU affinity setting. However, for Java based applications, many people feel

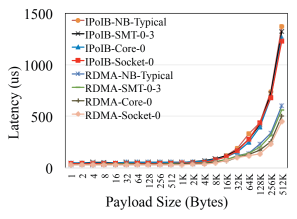


(a) IBM POWER8 ppc64le

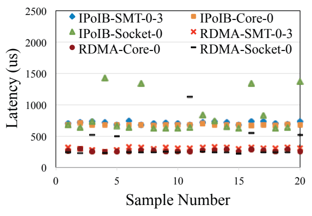


(b) Intel Xeon x86

Fig. 5: Variation Distributions of RDMA and IPoIB on POWER8 and Xeon x86 Platforms with OpenJDK 1.8



(a) Latency



(b) Variation Distributions

Fig. 6: Performance Comparison of RDMA and IPoIB on POWER8 with Different CPU Affinity Settings (IBM JDK 1.8)

it is better to let the OS and/or JVM make scheduling decisions for threads, because by default, JVM-based applications will have many internal threads for management, garbage collection, user-logic thread pools, etc. For example, in this simple RPC latency benchmark, we see more than 80 threads launched inside JVM for both server and client side on POWER8 platform. On Xeon x86 machines, there are also more than 40 Java threads launched for both server and client processes. So many threads running in a single JVM process will easily cause runtime noise on multi-core platforms and then further lead to performance variations. With these analyses, we decide to check what kind of impact of different CPU affinity settings will be for RDMA and IPoIB on OpenPOWER system.

As discussed in Section II, a POWER8 processor typically supports multiple SMT threads. On our platform, we have 8 SMT threads per single physical core. We then start trying with different cases of binding JVM threads (both client and server) from a single SMT thread to two SMT threads and then four SMT threads. In order to bind to a physical core, we bind all JVM threads to 8 successive SMT threads. For binding to a CPU Socket, we bind all JVM threads to 80 successive SMT threads. As shown in Figure 6(a), the legend 'IPoIB/RDMA-SMT-0-3' means we bind JVM threads to the first four SMT threads. The legend of 'IPoIB/RDMA-Core-0' means we bind JVM threads to the first physical core. The legend of 'IPoIB/RDMA-Socket-0' means we bind JVM threads to the cores on the first Socket. We compare these cases with the 'IPoIB/RDMA-NB-Typical' cases that show typical numbers we get on IPoIB and RDMA protocols without any CPU binding. Here, note that we do not show numbers for CPU affinity setting on single SMT thread and two SMT threads. This is because we find that those settings significantly degrade the performance of both IPoIB and RDMA, since running many threads on a small number of SMT threads brings a lot of resource contention, which essentially causes the performance overhead. We choose to bind threads to the cores on the first Socket, which is because the InfiniBand HCA card is attached to the first CPU Socket. Typically, the cores on the first CPU socket which are closer to InfiniBand HCA will give the application better communication performance.

From Figure 6(a), we see that proper CPU affinity settings improve the performance for both IPoIB and RDMA. For 512 KB message size, the best number of RDMA with binding threads to the first Socket is around 450 us while the typical number of RDMA is around 600 us, which means a good CPU affinity setting will bring 25% performance improvement. For IPoIB, similarly, we see the best number with CPU binding for 512 KB message size is 1,230 us while the typical number without binding is about 1,374 us, which means a good CPU affinity setting will gain 10% performance benefit. These results indicate that RDMA-based communication performance is heavily influenced by CPU affinity setting, which makes CPU binding become a desired feature for RDMA-based designs as compared to its importance to IPoIB performance.

We further conduct the variation distribution experiments with different binding schemes as the results shown in Figure 6(b). Here, we see that binding JVM threads to all the cores in the first CPU socket will give the best performance in terms of latency, however, it also shows large variations in some cases. This happens for both IPoIB and RDMA. By contrast, binding to the first physical core gives the best stability and reasonably good performance. Binding to the first four SMT threads also shows good stability but slightly worse performance. These numbers indicate that in order to get the best performance and stability, we should choose a proper CPU affinity setting policy, which needs to be tuned for different Big Data workloads.

IV. Proposed Accelerations for RDMA-based Communication Engine on OpenPOWER

In this section, we will describe the proposed accelerations for RDMA-based communication engine for Big Data workloads on OpenPOWER systems with InfiniBand.

A. Basic Ideas

As discussed in Section III, a proper CPU affinity setting may significantly influence the performance of RDMA-based Big Data processing middleware. We can choose to bind Java process in a coarse-grained manner as we have done in Section III through some operating system tools, such as `taskset` on Linux. However, the coarse-grained CPU affinity setting approach may not be a smart way for complex Big Data process workloads, since the Big Data processing middleware typically have many components and each of them has many internal threads. A naive CPU binding cannot do a better job than the OS or JVM. In our experiments, we also see that a bad CPU binding scheme hurts the whole job's throughput and execution time significantly. In this case, we ask the following vital question: should we just follow the traditional convention of letting the OS or JVM handle this complexity or find effective CPU binding policies?

However, as we have seen the performance characterization numbers in Section III, for IPoIB, letting JVM or OS do this work may be reasonable since the best binding can only have around 10% performance improvement compared to the default scheme. This is also why for the default Hadoop or Spark frameworks, there is no CPU affinity setting support in their out-of-box packages. However, for RDMA-based protocol, we have seen that a proper CPU affinity setting not only significantly improves the performance but also restrains the performance variation. This makes us believe that exploring CPU affinity for RDMA-based Big Data processing middleware is worthy. If this is a worthy task and a coarse-grained binding approach does not work out, we should explore the opportunities of designing fine-grained CPU binding method for Big Data frameworks. However, for the complex Hadoop- and Spark-like Big Data processing frameworks with possibly thousands of threads running in a single node, an efficient fine-grained threading affinity setting is not trivial.

Fortunately, one of our observations is that RDMA-based communication engine typically has less number of communication threads compared to default Sockets-based designs. For example, in RDMA-based Hadoop RPC [33], we see that with only 6 concurrent communication threads, RDMA-based Hadoop RPC can achieve more than 60% throughput improvement compared to the IPoIB scheme. Since our major goal is improving the communication performance and restraining the variation of it, we believe the fine-grained thread binding can be done only for communication threads. If we just need to bind a minimal number of communication threads to some specific cores, this will make the fine-grained thread affinity setting easier and it will also not influence the OS and JVM thread scheduling policies too much. Thus, it will minimize the side-effect on other compute threads.

B. Architecture Overview

With all of these observations, we propose our fine-grained CPU affinity aware designs in RDMA-based communicate engine for Big Data frameworks. As shown in Figure 7, our proposed accelerations target on Hadoop and Spark-based Big Data ecosystem. The important components include HDFS, MapReduce, Spark, Hadoop RPC, and many others. All the components will run on top of many Java processes and threads. The communication in default designs will go through Java Sockets library, TCP/IP stack, and IPoIB protocol over InfiniBand. In all RDMA-enhanced designs[1] for these components, the communication will go through RDMA over JNI layer, then interact with RDMA-based communication engine, which is running over native Verbs interface. Our proposed designs are mainly inside the RDMA-based communication engine, and the new proposed components include:

Device Selection Module: Since we want to bind communication threads to CPU cores, we need to support applications to select which InfiniBand device they are going to use, if there are multiple ones presented in the system. We will open the device based on the user's selection and also transfer this information to the locality detection module.

Locality Detection Module: Once we know which device is being selected, we need to detect its locality information. Basically, we need to know the opened device is attached to which CPU socket. This is important since if we bind communication

[1]We are using the packages in the High-Performance Big Data (HiBD) project. URL: http://hibd.cse.ohio-state.edu/

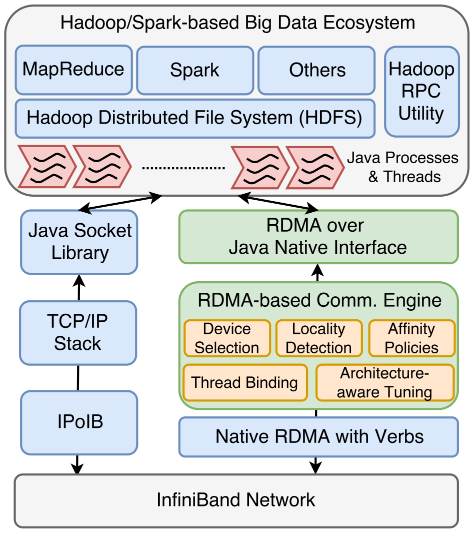


Fig. 7: Architecture Overview of Proposed Accelerations in RDMA-based Communication Engine

threads to CPU cores which are closer to InfiniBand HCA, it will improve the communication performance.

Affinity Policies: We propose multiple affinity policies that will be discussed in Section IV-C. We allow users to specify which policy they want to run with their applications, so that we can provide the flexibility for the end applications to have proper CPU bindings.

Thread Binding Module: Once the InfiniBand device is opened, locality information is detected, and the policy is chosen, we need to enforce the actual thread binding into the specific cores or SMT threads by the thread binding module. This module will interact with the OS to bind a particular Java thread to one or some specific SMT threads or cores. This module is working on the critical path of RDMA communication flow. Thus, all the upper layer RDMA-based communication threads will go through this module and will be pinned on some cores based on the selected policy. Since binding threads works on the critical path, we need to reduce such kind of interactions with the OS as much as possible. We further propose a thread binding cache in this module, which will store the current existing binding information. Every time, we will first check if the current thread entering the RDMA engine is already pinned or not. If it has been pinned earlier, we will avoid the repeated pinning. Otherwise, we will pin it and put it into the cache, so that next time if the same thread comes, it will be recognized in the cache and go through this module quickly.

C. Efficient CPU Affinity Setting Policies

We propose two broad categories of CPU-affinity setting policies in this paper:

(1) One Communication Thread to N SMT Threads on the Closer-to-HCA Socket (O2NTS): This policy will allow each communication thread run on N SMT threads on the cores of the POWER CPU on the socket closer to the InfiniBand HCA. Here, we allow users to tune N to adapt to the application's communication requirements. There are two special cases in

this category. (a) One Communication Thread to All Physical Cores on the Closer-to-HCA Socket (O2ACS): In this policy, we will bind each RDMA communication thread to all cores on the socket closest to the InfiniBand HCA, i.e., N=80. This scheme is motivated by the observations presented in Section III, where we see such binding can potentially deliver the best latency numbers; (b) One Communication Thread to One Physical Core on the Closer-to-HCA Socket (O2OCS): Based on the numbers presented in Section III, binding a communication thread to a single physical core on the socket closer to the IB HCA, i.e., N=8, will also bring performance benefits as well as a reduction for the performance variation. With this policy, we can bind the RDMA communication threads to physical cores on the closer-to-IB-HCA CPU in a round-robin fashion. Thus, we make the best use of the available thread density and concurrency.

(2) One Communication Thread to N SMT Threads on a CPU (O2NTC): This policy allows the RDMA communication thread be pinned to N SMT threads available on any of the POWER cores. Unlike the O2NTS policy, the communication threads can be bound to any of the two sockets (i.e., either socket 0 or socket 1) and not just the socket closer to the InfiniBand HCA. Under this general CPU-affinity policy, we have: One Communication Thread to One Physical Core on the CPU (O2OCC). For O2OCC, we bind one RDMA communication thread to one physical core on the POWER8 CPU, i.e., N=8. This policy will be beneficial if there are a lot of upper-layer communication threads and we want to load balance them across all the cores available on the CPU.

D. Architecture-aware Tuning

As we see that the POWER architecture is quite different than x86, especially with many physical cores and SMT threads available, our experience shows that it is very important to perform POWER architecture-aware tuning for the RDMA engine, in which the default parameters were well tuned for x86 architecture but they may not the desired ones for POWER platforms. In our design, we first support POWER architecture detection in our RDMA engine and then, our library will automatically choose the tuned parameters for POWER architecture rather than the ones for x86. Currently, these parameters are tuned in advance based on our testbeds. We also support users to easily overwrite these values through environment variables if they find the chosen parameter values are not performing well on their POWER machines. From our early experience, we observe that there are many performance sensitive parameters to achieve good performance, such as chunk size per RDMA operation, the number of handler threads for data transferring, RDMA buffer pool size, etc. This also implies that to achieve optimal performance for RDMA-based designs on POWER machines is a big challenge, which opens up a lot of future work opportunities.

V. Performance Evaluation

In this section, we present the detailed performance evaluations of our proposed design and its impact on Apache Hadoop and Apache Spark workloads.

A. Experimental Setup

In the following experiments, we use the OpenPOWER cluster as mentioned in Table II. We use up to four OpenPOWER machines. Since OpenPOWER system is still quite new, we do not have access to a large-scale OpenPOWER cluster. We use IBM JDK 1.8 in our experiments. Apache Hadoop 2.7.3 and Apache Spark 2.1.0 are used as default Hadoop and Spark stacks to compare with. We integrate our proposed designs with RDMA-Hadoop 1.1.0 and RDMA-Spark 0.9.4, which are derivatives of Apache Hadoop 2.7.3 and Apache Spark 2.1.0, respectively. In the following experiments, the legend of 'IPoIB' means we run the default package with IPoIB protocol. The legend of 'RDMA-Def' means we run the RDMA-Hadoop and RDMA-Spark default designs over RDMA protocol. The legend of 'RDMA-Opt' means we run RDMA-Hadoop and RDMA-Spark with our proposed designs over RDMA protocol.

We choose various workloads as listed in Table III. These workloads aim to cover communication intensive workloads, I/O intensive workloads, and comprehensive workloads (having requirements on communication, computation, and I/O), which can run on top of both Hadoop and Spark. Another reason of choosing the workloads or benchmarks listed in Table III is because these benchmarks can run with the targeted platform in an isolated manner. When running them, we do not need to involve many other components, which will avoid the performance influence from other components.

B. Performance Evaluation with OHB Hadoop RPC Benchmarks

In this subsection, we evaluate IPoIB-based Hadoop, RDMA-Hadoop, and RDMA-Hadoop with our proposed accelerations on the OpenPOWER cluster. Figure 8 shows the performance evaluation with OHB Hadoop RPC latency benchmark results. The legend of 'IPoIB-Def' means the typical numbers we get with IPoIB-based Hadoop without any manual CPU binding. The legend of 'IPoIB-Manual-Best-Binding' means the best numbers we can get after we try with different binding schemes for IPoIB-based Hadoop. Similarly, we show the corresponding numbers for 'RDMA-Def' and 'RDMA-Manual-Best-Binding'. We also take all the numbers with the proposed five different CPU affinity setting policies as we discussed in Section IV-C. These numbers are presented under the legends of 'RDMA + Policy Name'.

From these results, we first see that all the RDMA-based schemes are obviously performing much better than IPoIB. The best number with RDMA shows 2.73X performance speedup compared to the best number we can get with IPoIB. Secondly, we find that with our proposed fine-grained thread binding schemes, we are able to achieve similar performance as the best binding scheme by manually. For example, the best-binding scheme by manually can achieve 450 us latency for 512 KB message size, while the 'RDMA-O2ACS' scheme is able to get around 453 us latency. In the 'RDMA-O2ACS' scheme, all the communication threads will be allowed to run on all the cores which are on the close-to-HCA socket. These results also match with the experiments we have done in Section III.

In addition, we find 'RDMA-O2NTS (N=1)' and 'RDMA-O2NTC (N=1)' are showing slightly worse performance com-

TABLE III: Workload Description and Category

Workload	Description	Category	Targeted Platform
OHB-RPC-Latency [37]	OSU HiBD Microbenchmark for evaluating Hadoop RPC latency	Communication-Intensive	Hadoop
TestDFSIO	Hadoop Distributed File System Benchmark to evaluate throughput and average I/O rate	I/O-Intensive	Hadoop
GroupBy	A Spark benchmark to group the values for each key in the RDD into a single sequence	Comprehensive Benchmark	Spark
SortBy	A Spark benchmark to sort the the RDD by key	Comprehensive Benchmark	Spark

pared to other RDMA binding schemes, which indicates that if we bind each communication thread with one SMT thread, we can not utilize the CPU cores efficiently. Rather than these, if we bind the communication thread with one physical core (i.e., 8 SMT threads) or one Socket (i.e., 80 SMT threads; closer to HCA), the performance is very close to the best binding scheme by manually. In the following experiments with Hadoop and Spark workloads, we choose 'RDMA-O2ACS' as the default policy.

Regarding performance variations, since we only bind communication threads to some cores, there still have some small variations due to Java garbage collection threads and other JVM management threads running over different cores. We do not want to bind those threads because, for complex application scenarios, those threads need to take care of the memory garbage collection and other management tasks in the whole JVM. It is better to let the OS and JVM make the scheduling decision for them.

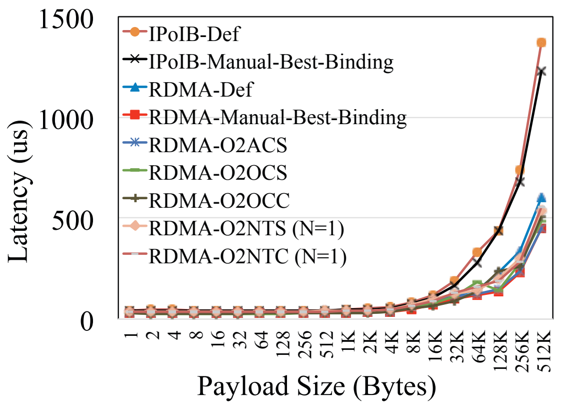
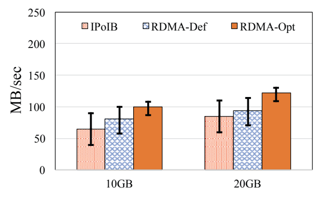


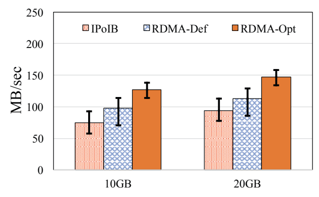
Fig. 8: Performance Evaluation with OHB Hadoop RPC Benchmark

C. Performance Evaluation with Hadoop Benchmarks

In this subsection, IPoIB-based Hadoop, RDMA-Def Hadoop, and RDMA-Opt Hadoop are evaluated with Hadoop TestDFSIO Write Benchmark on the OpenPOWER cluster listed in Table II. For each experiment, four nodes are used as DataNodes, and one of them is configured as NameNode as well. For each TestDFSIO Write benchmark, 64 mappers and 32 reducers are used. Figure 9 illustrates the performance differences and highlights the performance variations (shown as error bars) of different Hadoops on the OpenPOWER cluster. Note that the tested data size does not go beyond 20 GB, which is mainly because the available local disk space (around 30 – 40 GB) is quite small on each of these OpenPOWER machines. Due to the default three-replica setting, we could only run 10 GB and 20 GB tests for both IPoIB and RDMA designs on this cluster. We report throughput and average IO rate for Hadoop TestDFSIO Write benchmark.



(a) Throughput



(b) Average IO Rate

Fig. 9: Performance Evaluation with Hadoop TestDFSIO Write

Throughput: As shown in Figure 9(a), the RDMA-Opt design can achieve up to 29.37% improvement compared to the RDMA-Def for Hadoop TestDFSIO Write, and outperforms IPoIB-based Hadoop by up to 53.73%. Meanwhile, we observe that RDMA-Opt is the most stable scheme that has a variation of about -13.04% to +7.99%, as compared to IPoIB's variation of about -24.90% to +25.07% and the RDMA-Def's performance variation of around -23.45% to +19.38%.

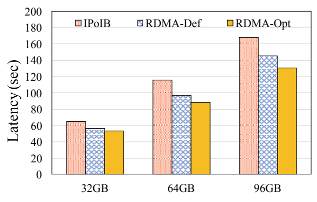
Average IO Rate: From average IO rate perspective, the RDMA-Opt design improves performance by about 29.97% over RDMA-Def and up to 70.60% as compared to default design running over IPoIB. The performance variations for IPoIB, RDMA-Def, and RDMA-Opt are -16.79% to +18.46%, -27.13% to +15.87%, and -13.35% to +10.90%, respectively.

From these numbers, we see that our RDMA-Opt design, with proper communication thread binding and architecture-aware tuning, can effectively and efficiently exploit the OpenPOWER clusters, while dramatically improve performance as well as reduce performance variation.

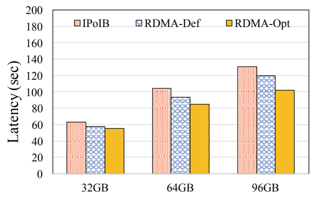
D. Performance Evaluation with Spark Benchmarks

In this section, we evaluate IPoIB-based Spark, RDMA-Def Spark, and RDMA-Opt Spark with Spark SortBy and GroupBy benchmarks on the OpenPOWER cluster. For all experiments presented in this section, the Spark cluster is deployed on four nodes; with one of the four nodes acting as Spark Master and the other three run the Spark Workers. For each Spark run, the number of maps and reduces are configured as 480 to saturate all CPU cores on the OpenPOWER-based cluster.

The performance results of different Spark designs are shown in Figure 10. From Figure 10(a), we observe that the RDMA-Opt design can demonstrate up to 23.52% performance gain over default Spark running over IPoIB, and up to 10.48% performance improvement as compared the default RDMA-based Spark design, while running the SortBy benchmark. Similarly, Figure 10(b) illustrates that for the Spark GroupBy benchmark, the RDMA-Opt design outperforms the IPoIB and the RDMA-Def by 22.15% and 14.83%, respectively. These results show that for Spark workloads, our proposed enhanced schemes can also perform better than default IPoIB and RDMA schemes on POWER architecture.



(a) Spark SortBy Benchmark



(b) Spark GroupBy Benchmark

Fig. 10: Performance Evaluation with Spark Benchmarks

VI. Related Work

In this section, we discuss related work in the following two categories.

HPC Middleware on OpenPOWER: Softwares and middlewares that are primarily employed by scientific applications, such as MPI [10, 38], that can exploit and deliver optimal performance on HPC clusters, including, IBM POWER systems has been a popular topic among researchers. Performance evaluations of OpenMP or a hybrid OpenMP+MPI-based parallelization approaches for HPC scientific applications on POWER8 architecture are presented in [19, 20, 45]. Research work such as [32, 47] explore ways to improve the scalability and performance of MPI libraries on POWER8 InfiniBand clusters, along with GPU-to-GPU communication. Similarly, to leverage the capabilities of the OpenPOWER architecture, researchers have proposed new and enhanced collective communication algorithms to accelerate MPI-based scientific applications [21]. In addition to MPI-based HPC applications, IBM OpenPOWER clusters are enabling advances in medical research through accelerating large-scale and data-intensive genomics analytics [4, 8].

High Performance Data Analytics on OpenPOWER: Recently, substantial research efforts are being directed towards exploring means to exploit the high thread density and concurrency of OpenPOWER for accelerating Big Data processing middleware. Towards this, IBM has introduced a free, open source distribution of Apache Hadoop and Spark to work seamlessly on IBM systems, including, OpenPOWER clusters [7]. With the increasing popularity of Apache Spark for designing Big Data applications, efforts are being made to employ IBM POWER systems to accelerate Spark workloads [1, 16, 22]. Studies have been dedicated towards understanding the performance of different Spark workloads on OpenPOWER clusters [42], and tuning Spark/JVM parameters to fully leverage the memory bandwidth, cache structure, thread density, etc., enabled by the OpenPOWER architecture [2]. Research studies such as [29] have been devoted to proposing prediction-based frameworks that can auto-adjust the thread count in SMT cores on POWER8 processors for obtaining optimal performance for various Spark-based Big Data workloads. In addition to traditional Big Data workloads, OpenPOWER technologies such as CAPI and NVlink coupled with GPU-/FPGA-enabled and high-bandwidth network-connected clusters present a desirable end-to-end solution for training and classification of datasets in machine and deep learning frameworks [9, 11]. In contrast, this paper focuses on analyzing and accelerating Apache Hadoop and Spark frameworks that have been redesigned with RDMA for InfiniBand/RoCE-enabled HPC clusters [34], in order to enable optimal performance on OpenPOWER clusters.

VII. Conclusion

In this paper, we first illustrate extensive performance characteristics on RDMA-based Hadoop RPC engine over an 'POWER8 + InfiniBand' platform. Our performance characterization shows that existing RDMA-based communication engine for Big Data processing middleware originally designed for x86 architectures still outperforms the default designs running over IPoIB protocol. In the meantime, we observe that the performance of both IPoIB and RDMA on the OpenPOWER platforms can vary a lot. Based on the analysis, we propose several designs to enable efficient CPU affinity setting policies and architecture-aware tuning inside RDMA-based communication engines for both Hadoop and Spark to exploit the OpenPOWER multi-core platforms. Our evaluation results show that our proposed RDMA-Opt Hadoop design can achieve up to 2.73X speedup compared to IPoIB-based Hadoop, up to 29.37% throughput performance improvement compared to the RDMA-Def Hadoop. Our RDMA-Opt Spark design can gain up to 23.52% performance improvement compared to the IPoIB-based Spark, up to 14.83% speedup compared to the RDMA-Def Spark. We also see that our proposed designs for Hadoop and Spark can reduce the performance variations for workloads running on the OpenPOWER platforms. Overall, our proposed designs achieve higher and more stable performance on the OpenPOWER systems with native RDMA-based communication engines for popular Big Data processing middleware such as Apache Hadoop and Spark.

In the future, we plan to investigate further on accelerating Big Data processing middleware on the OpenPOWER platforms and propose new designs along these directions. We also plan to make all of our designs available through RDMA-based Hadoop and Spark packages.

References

[1] "Apache Spark on OpenPOWER," OpenPOWER Summit 2016. [Online]. Available: https://openpowerfoundation.org/presentations/apache-spark-on-openpower/

[2] "Apache Spark Runs 2X Faster on IBM's POWER8," https://www.ibm.com/developerworks/community/blogs/f0f3cd83-63c2-4744-9021-9ff31e7004a9/entry/Apache_Spark_Runs_2X_Faster_on_IBM_s_POWER8?lang=en.

[3] "Coherent Accelerator Processor Interface (CAPI)," https://developer.ibm.com/linuxonpower/capi/.

[4] "Genome Folding and POWER8: Accelerating Insight and Discovery in Medical Research," https://openpowerfoundation.org/blogs/genome-folding-with-power8/.

[5] "Hadoop* Clusters Built on 10 Gigabit Ethernet," https://www.arista.com/assets/data/pdf/Whitepapers/Hadoop_WP_final.pdf.

[6] "IBM Crail," http://www.crail.io/.

[7] "IBM Open Platform for Apache Hadoop and Apache Spark," https://www.ibm.com/us-en/marketplace/ibm-open-platform.

[8] "IBM POWER8 HPC System Accelerates Genomics Analysis with SMT8 Multithreading," OpenPOWER Summit 2016. [Online]. Available: https://openpowerfoundation.org/presentations/ibm-power8-hpc-system-accelerates-genomics-analysis-with-smt8-\multithreading/

[9] "IBM PowerAI," https://www.ibm.com/blogs/systems/powerai-the-worlds-fastest-deep-learning-solution-among-leading-enterprise-servers/.

[10] "IBM Spectrum MPI: Accelerating High-Performance Application Parallelization," https://www-03.ibm.com/systems/spectrum-computing/products/mpi/.

[11] "Machine and Deep Learning on Power Systems," OpenPOWER Summit 2016. [Online]. Available: https://openpowerfoundation.org/presentations/

[12] "Mellanox Supercharges Spark, Delivers Industrys First 100TB Spark SQL Benchmark," http://www.mellanox.com/blog/2016/11/mellanox-supercharges-spark-delivers-industrys-first-100tb-sql-benchmark/.

[13] "OpenPOWER," https://openpowerfoundation.org/.

[14] "POWER8 – The First OpenPOWER Processor," https://openpowerfoundation.org/blogs/power8-the-first-openpower-processor/.

[15] "Tencent Galvanizes OpenPower with Mellanox 100GbE Network, Breaking World Records!" http://www.mellanox.com/blog/2016/11/tencent-galvanizes-openpower-with-mellanox/.

[16] "Bringing Analytics and Science Closer Together with Spark," OpenPOWER Academic Discussion Group Workshop, 2015.

[17] "Mellanox EDR InfiniBand Speeds OpenPOWER Platforms," https://insidehpc.com/2015/11/mellanox-edr-infiniband-speeds-openpower-platforms/, 2015.

[18] "IDCs Latest Forecast for the HPC Market: 2016 is Looking Good," https://www.top500.org/news/idcs-latest-forecast-for-the-hpc-market-2016-is-looking-good/, 2016.

[19] A. V. Adinetz, P. F. Baumeister, H. Böttiger, T. Hater, T. Maurer, D. Pleiter, W. Schenck, and S. F. Schifano, "Performance Evaluation of Scientific Applications on POWER8," in *International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*. Springer, 2014, pp. 24–45.

[20] A. Berreth, B. Risio, M. Bühler, B. Anlauf, and P. Vezolle, "Performance of the 3D Combustion Simulation Code RECOM®-AIOLOS on IBM® POWER8® Architecture," in *International Conference on High Performance Computing*. Springer, 2016, pp. 286–292.

[21] S. Chakraborty, H. Subramoni, and D. K. Panda, "Contention-Aware Kernel-Assisted MPI Collectives for Multi-/Many-core Systems," in *2017 IEEE International Conference on Cluster Computing (Cluster '17)*, Sept 2017.

[22] T. Chiba and T. Onodera, "Workload Characterization and Optimization of TPC-H Queries on Apache Spark," in *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2016, pp. 112–121.

[23] A. Dragojević, D. Narayanan, O. Hodson, and M. Castro, "FaRM: Fast Remote Memory," in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, 2014, pp. 401–414.

[24] S. Hefty, "Rsockets," in *2012 OpenFabrics International Workshop*, 2012.

[25] J. Huang, X. Ouyang, J. Jose, M. W. ur Rahman, H. Wang, M. Luo, H. Subramoni, C. Murthy, and D. K. Panda, "High-Performance Design of HBase with RDMA over InfiniBand," in *IEEE 26th International Parallel and Distributed Processing Symposium (IPDPS)*, May 2012, pp. 774–785.

[26] InfiniBand Trade Association, "Infiniband," http://www.infinibandta.org/.

[27] 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.

[28] N. S. Islam, X. Lu, M. W.-u. Rahman, and D. K. D. Panda, "SOR-HDFS: A SEDA-based Approach to Maximize Overlapping in RDMA-enhanced HDFS," in *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*, ser. HPDC '14. New York, NY, USA: ACM, 2014, pp. 261–264.

[29] Z. Jia, C. Xue, G. Chen, J. Zhan, L. Zhang, Y. Lin, and P. Hofstee, "Auto-tuning Spark Big Data Workloads on POWER8: Prediction-Based Dynamic SMT Threading," in *Proceedings of the 2016 International Conference on Parallel Architectures and Compilation*, ser. PACT '16. ACM, 2016, pp. 387–400.

[30] A. Kalia, M. Kaminsky, and D. G. Andersen, "Using RDMA Efficiently for Key-Value Services," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 295–306.

[31] T. S. Karkhanis and J. E. Moreira, *IBM Power Architecture*. Boston, MA: Springer US, 2011, pp. 900–907.

[32] S. Kumar, R. Blackmore, S. Sharkawi, N. J. K. A., A. Mamidala, and T. J. C. Ward, "Optimization of Message Passing Services on POWER8 InfiniBand Clusters," in *Proceedings of the 23rd European MPI Users' Group Meeting*, ser. EuroMPI 2016. New York, NY, USA: ACM, 2016, pp. 158–166.

[33] X. Lu, N. S. Islam, M. W. Rahman, J. Jose, H. Subramoni, H. Wang, and D. K. Panda, "High-Performance Design of Hadoop RPC with RDMA over InfiniBand," in *The Proceedings of IEEE 42nd International Conference on Parallel Processing (ICPP)*, France, October 2013.

[34] X. Lu, M. Rahman, N. Islam, D. Shankar, and D. Panda, "Accelerating Spark with RDMA for Big Data Processing: Early Experiences," in *2014 IEEE 22nd Annual Symposium on High-Performance Interconnects (HOTI)*, 2014.

[35] X. Lu, D. Shankar, S. Gugnani, and D. K. D. Panda, "High-Performance Design of Apache Spark with RDMA and Its Benefits on Various Workloads," in *2016 IEEE International Conference on Big Data (BigData '16)*. IEEE, 2016, pp. 253–262.

[36] X. Lu, D. Shankar, S. Gugnani, H. Subramoni, and D. K. Panda, "Impact of HPC Cloud Networking Technologies on Accelerating Hadoop RPC and HBase," in *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2016, pp. 310–317.

[37] X. Lu, M. Wasi-ur Rahman, N. S. Islam, and D. K. D. Panda, "A Micro-benchmark Suite for Evaluating Hadoop RPC on High-Performance Networks," in *Workshop on Big Data Benchmarks (WBDB '13)*. Springer, 2013, pp. 32–42.

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

[39] OpenFabrics Alliance, http://www.openfabrics.org/.

[40] OSU NBCL, "The High-Performance Big Data Project," http://hibd.cse.ohio-state.edu.

[41] N. Poggi, D. Carrera, A. Call, S. Mendoza, Y. Becerra, J. Torres, E. Ayguad, F. Gagliardi, J. Labarta, R. Reinauer, N. Vujic, D. Green, and J. Blakeley, "ALOJA: A Systematic Study of Hadoop Deployment Variables to Enable Automated Characterization of Cost-Effectiveness," in *2014 IEEE International Conference on Big Data (Big Data)*, Oct 2014, pp. 905–913.

[42] X. Que, L. Schneidenbach, F. Checconi, C. H. Costa, and D. Buono, "Performance Analysis of Spark/GraphX on POWER8 Cluster," in *International Conference on High Performance Computing*. Springer, 2016, pp. 268–285.

[43] M. W. Rahman, N. S. Islam, X. Lu, J. Jose, H. Subramoni, H. Wang, and D. K. Panda, "High-Performance RDMA-based Design of Hadoop MapReduce over InfiniBand," in *The Proceedings of International Workshop on High Performance Data Intensive Computing (HPDIC), in conjunction with IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Boston, May 2013.

[44] M. W. Rahman, X. Lu, N. S. Islam, and D. K. Panda, "HOMR: A Hybrid Approach to Exploit Maximum Overlapping in MapReduce over High Performance Interconnects," in *International Conference on Supercomputing (ICS)*, Munich, Germany, 2014.

[45] I. Z. Reguly, A.-K. Keita, R. Zurob, and M. B. Giles, "High-Performance Computing on the IBM Power8 Platform," in *International Conference on High Performance Computing*. Springer, 2016, pp. 235–254.

[46] D. Shankar, X. Lu, N. Islam, M. Wasi-Ur-Rahman, and D. K. Panda, "High-Performance Hybrid Key-Value Store on Modern Clusters with RDMA Interconnects and SSDs: Non-blocking Extensions, Designs, and Benefits," in *2016 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2016, pp. 393–402.

[47] C. D. Sudheer and A. Srinivasan, "Efficient Barrier Implementation on the POWER8 Processor," in *2015 IEEE 22nd International Conference on High Performance Computing (HiPC)*. IEEE, 2015, pp. 165–173.

[48] B. Thompto, "POWER9 Processor for the Cognitive Era," in *2016 IEEE Hot Chips 28 Symposium (HCS)*. IEEE, 2016, pp. 1–19.

[49] Y. Degani, "Accelerating Apache Spark with RDMA," 13th Annual OpenFabrics Alliance Workshop, 2017. [Online]. Available: https://www.openfabrics.org/images/eventpresos/2017presentations/108_Apache_YDegani.pdf