BenchCouncil Transactions on Benchmarks, Standards and Evaluations 2 (2022) 100074

Contents lists available at ScienceDirect

BenchCouncil Transactions on Benchmarks, Standards and Evaluations

journal homepage: www.keaipublishing.com/en/journals/benchcouncil-transactions-on-benchmarks-standards-and-evaluations/

Research Article

Understanding hot interconnects with an extensive benchmark survey

Yuke Li [a], Hao Qi [a], Gang Lu [b], Feng Jin [b], Yanfei Guo [c], Xiaoyi Lu [a,*]

[a] University of California, Merced, 5200 North Lake Rd., Merced, 95343, CA, USA
[b] Tencent, No. 33, Haitian Second Road, Shenzhen, 518054, Guangdong, China
[c] Argonne National Laboratory, 9700 S. Cass Avenue, Lemont, 60439, IL, USA

ARTICLE INFO

Keywords:
Benchmarks
Interconnects
RDMA

ABSTRACT

Understanding the designs and performance characterizations of hot interconnects on modern data center and high-performance computing (HPC) clusters is a fruitful research topic in recent years. The rapid and continuous growth of high-bandwidth and low-latency communication requirements for various types of data center and HPC applications (such as big data, deep learning, and microservices) has been pushing the envelope of advanced interconnect designs. We believe this is high time to investigate the performance characterizations of representative hot interconnects with different benchmarks. Hence, this paper presents an extensive survey of state-of-the-art hot interconnects on data center and HPC clusters and the associated representative benchmarks to help the community to better understand modern interconnects. In addition, we characterize these interconnects by the related benchmarks under different application scenarios. We provide our perspectives on benchmarking data center interconnects based on our survey, experiments, and results.

1. Introduction

The scales of data center and high-performance computing (HPC) clusters grow rapidly with the increasingly large volume of data and the high demand for distributed computing capabilities [1]. This trend has led to various designs of modern data center interconnects and made their performance characterizations a rewarding research topic. To continuously improve the performance and scalability of data movement or communication across a large number of nodes in modern data center or HPC clusters, different types of advanced interconnects have been designed to meet the requirements of high-bandwidth and low-latency communications in popular data center applications, such as deep learning, big data, microservices, etc.

To upgrade the conventional Ethernet (~10 Gbps) network and accelerate the efficiency of data center applications, hardware vendors have demonstrated multiple types of advanced data center interconnects. For example, NVIDIA (Mellanox) has produced 200 Gbps InfiniBand (IB) [2] with well-optimized Remote Direct Memory Access (RDMA) subsystems to speedup the inter-node communication in applications. Cray has the Slingshot interconnect [3] and the Aries interconnect [4] as high-speed interconnects for modern HPC systems. RIKEN (Japanese Institute of Physical and Chemical Research) and Fujitsu developed the Tofu interconnect [5] family to be equipped on their designed supercomputers. Meanwhile, the Ethernet network speed has improved from 10 Gbps to 100 Gbps [6] and even above [7] during the decades of development.

With the trend of hardware evolution and the new interconnects being created, there are several issues that the application developers need to pay attention to. With the hardware upgrading, the developers need to re-evaluate the performance of different generations of hardware to design the proper systems software based on the improved data transfer rates. Also, many new interconnects are emerging with the development of novel hardware features. These features may potentially impact application performance and need to be systematically investigated.

On the other hand, different types of data center applications represent various performance characterizations, like HPC workloads, deep learning training and inferences, big data analytics, and cloud-based microservice. The impacts of new interconnects on these different workloads should be evaluated separately and carefully. Therefore, we believe this is high time to investigate the performance characterizations of modern data centers and HPC interconnects via standard benchmarking experiments under different application scenarios. This observation motivates us to extensively survey hot interconnects on modern data centers and HPC clusters and the associated representative benchmarks to help the community better understand these advanced interconnects.

There exist some surveys to summarize benchmarking experiences with different workloads. For example, Han et al. [8] surveyed ten big data benchmarks to discuss benchmarking challenges. Zhang et al. [9]

* Corresponding author.
E-mail addresses: yli304@ucmerced.edu (Y. Li), hqi6@ucmerced.edu (H. Qi), gateslu@tencent.com (G. Lu), ronyjin@tencent.com (F. Jin), yguo@anl.gov (Y. Guo), xiaoyi.lu@ucmerced.edu (X. Lu).

https://doi.org/10.1016/j.tbench.2022.100074
Received 19 September 2022; Received in revised form 21 October 2022; Accepted 21 October 2022
Available online 28 October 2022

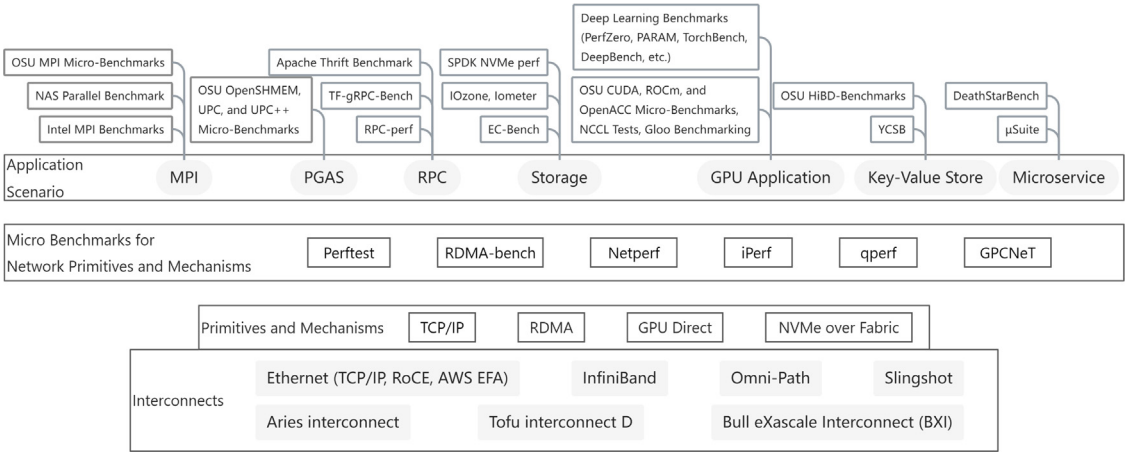


Fig. 1. An overview of data center interconnects and benchmarks.

investigated fourteen deep learning benchmarks. Zhou et al. [10] discussed seven microservice benchmarks. Gao et al. [11] compared fifteen big data and AI (Artificial Intelligence) benchmarks. However, we did not find such a survey that can extensively cover a broad range of the latest advanced interconnects in modern data centers and the associated representative benchmarks for different application scenarios. Therefore, this paper addresses the need to survey different hot interconnects deployed in modern data centers and the corresponding benchmarks to expose their performance characteristics.

Fig. 1 shows an overview of this paper's surveying scope, including various kinds of hot interconnects and the associated representative and popular benchmarks in the community. The following sections will introduce each component in Fig. 1 with a bottom-up approach. In Section 2, we survey the features and characteristics of hot interconnects in modern data centers and HPC clusters. In Section 3, we survey well-used micro benchmarks for evaluating these hot interconnects with their network primitives and mechanisms. Section 4 will survey application-level benchmarks with diverse evaluation granularity, as shown in Fig. 1. In Section 5, we choose several representative benchmarks, which include Netperf [12], Perftest [13], and OSU Micro-Benchmarks (OMB) [14] for MPI (Message Passing Interface) [15] and PGAS (Partitioned Global Address Space) [16] applications, and interconnects, which include IB, Omni-Path [17], and Ethernet to run experiments. We present the results to show performance characterizations of these hot interconnects as examples or reference numbers. Section 6 will discuss some of our observations and perspectives on benchmarking data center interconnects based on our survey, experiments, and results. Section 7 discusses more related studies and Section 8 concludes the paper.

The main contributions of this paper are as follows:

- We perform an extensive survey on advanced hot interconnects in current-generation and emerging data centers and HPC clusters.
- We also comprehensively survey the associated representative benchmarks from both micro benchmarking and application-level benchmarking perspectives.
- We perform a set of benchmarking experiments on real interconnects hardware with well-used benchmarks and discuss their performance characterizations.
- We share our observations on improvable aspects of existing benchmarks, such as performance stability, reference number, experimental instructions, etc., to help the community to design better ones.

2. Overview of modern interconnects

As an indispensable part of HPC and data center systems, interconnects play an essential role in achieving higher scalability and performance for modern clusters. In recent years, the community has witnessed the development of conventional interconnects like Ethernet and InfiniBand, and the birth of proprietary interconnects such as Fugaku Tofu [5] and BXI (Bull eXascale Interconnect) [18]. This section will briefly overview some representative state-of-the-art modern interconnects, and their features [1]. After we go through these interconnects one by one, Table 1 shows a brief comparison of these hot interconnects.

2.1. Ethernet

Ethernet is one of the most traditionally utilized interconnects for HPC and data center clusters. At the early stage, 1 Gb/s Ethernet (1-GigE) was widely used. However, with the advancement of CPU performance and I/O speed, the 1-GigE has become the bottleneck. With the demand for higher bandwidth and data transfer rate, Ethernet with 10-GigE, 25-GigE, 50-GigE, and even 100-GigE, has been developed. As of June 2022, 25-GigE is the most widely used interconnect in the Top500 list, and the Ethernet interconnect family is the majority in the list, taking up nearly 50% [19].

Taken the advantages of RDMA, RDMA over Converged Ethernet (RoCE) [20] is developed, which is a network protocol that allows RDMA to operate over Ethernet networks. RoCE is designed to support RDMA over Ethernet on layer 2 networks, and its extended version RoCE v2 enables transportation on layer 3 networks. Traditionally, Ethernet has left the congestion control to the TCP (Transmission Control Protocol) layer. With the development, the first algorithm proposed for the Ethernet network is pause frame [21] in 1996. Congestion control on RoCE uses an extension to the TCP/IP protocol called ECN (Explicit Congestion Notification) [22]. Other techniques, such as the QCN (Quantized Congestion Notification) [23], were developed afterward. Both traditional Ethernet and RoCE are available for various interconnect topologies. In 2019, Amazon announced EFA (Elastic Fabric Adapter) [24] for its EC2 (Elastic Compute Cloud) instance. The libfabric [25] interface on EFA provides up to 100 Gbps speed and reduces overhead with techniques like operating system bypass.

2.2. InfiniBand

Provided by NVIDIA, InfiniBand (IB) is an industry-standard switch fabric and the second most popular interconnect family in the Top500 list [19]. As of June 2022, 32.4% of the Top500 clusters are interconnected by IB, especially for Top10 clusters such as Summit [26] and Sierra [27]. Besides the higher bandwidth (up to 400 Gbps) and lower latency (<1 µs), IB also supports advanced features like RDMA, which allows the software to read/write data from/to the memory in remote

Table 1
Comparison of interconnects.

Name	Ethernet			InfiniBand	Omni-Path	Slingshot	Aries	TofuD	BXI
	25–100 Gbps	200–400 Gbps	RoCE						
Manufacturer	Many	Many	Many	NVIDIA/Mellanox	Intel/Cornelis	Cray	Cray	Fujitsu	Atos
Commodity	Public	Public	Public	Public	Public	Proprietary	Proprietary	Proprietary	Proprietary
Unidirectional Bandwidth (Gbps)	25–100 [37]	200–400 [37]	100 [38]	400 [39]	100 [37]	200 [37]	40 [4]	56 [40]	100 [37]
End to End Latency (μs)	10–30	N/A	~1 [37]	<1 [37]	<1 [37]	<2 [37]	~1 [4]	0.5–1 [5]	<1 [37]
Congestion Control	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Topology	Various	Various	Various	Fat-tree, Dragonfly+	Fat-tree	Dragonfly	Dragonfly	Torus	Various
RDMA	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Year	2014	2017	2010	1999	2015	2019	2012	2018	2015

nodes without any CPU involvement from the remote side. IB provides reliable or unreliable, and connected or datagram data transport types [28]. The reliable transport can guarantee the packet delivery in order but it spends extra time to wait for the acknowledgment from the receiving side. The unreliable transport cannot ensure the packet is received, but it does not need extra time for waiting the acknowledgment. The queue pairs for connected transports are connected in the one-to-one mapping, while the queue pairs for datagram transports are connected in the one-to-any mapping. The connected transport is more suitable for applications with a small number of connections. The datagram transport usually performs better in large-scale applications because fewer connection contexts need to be maintained in memory [29].

Specifically, RDMA-capable networks (like InfiniBand) typically support four types of transport modes: Reliable Connection (RC), Reliable Datagram (RD), Unreliable Connection (UC), and Unreliable Datagram (UD). `SEND` and `RECV` operations are supported by all modes, while the `RDMA WRITE` operation is unsupported by UD, and the `RDMA READ` operation is unsupported by UD and UC. The most commonly used network topology for IB is fat-tree [30], but it also supports other topologies like dragonfly+ [31]. The IB standard includes a congestion control mechanism to detect and resolve congestion by using two relay messages: FECN (Forward Explicit Congestion Notification) [32] and BECN (Backward Explicit Congestion Notification) [33]. When applying IB to GPU, CUDA 5.0 first introduced GDR (GPUDirect RDMA) [34]. GDR allows IB adapters to directly access the GPU memory while also bypassing the host. GDR can significantly increase data communication performance among GPUs, which further benefits the increasing number of redesigned classical HPC and machine/deep learning applications.

2.3. Omni-Path

Omni-Path was first released by Intel in 2015 as a part of Intel's Scalable System Framework with the purpose of increasing HPC workload scalability and aiming for low communication latency, low power consumption, and high throughput [17]. Omni-Path mainly includes the network card, switch, and network manager components. It is built on Intel technology with multiple features, such as traffic flow optimization and packet integrity protection. It is mainly designed to support fat-tree topology, and its CCA (Congestion Control Architecture) has been updated continuously since its first release. The first generation Omni-Path delivers 100 Gbps bandwidth per port and is integrated into some CPU architectures like Skylake and Knights Landing (KNL) [35]. Although Intel stopped the development of the second-generation Omni-Path in 2019, it still takes 7.8% of Top500 clusters as of June 2022 [19]. In late 2020, Intel announced its spin-off to Cornelis Networks [36] to continue the business as a successor to the Omni-Path product.

2.4. Slingshot

In 2019, Cray launched its new generation of HPC interconnect technology called Slingshot [3]. Slingshot uses protocols on standard Ethernet while also being compatible with proprietary HPC networks when needed. It offers key features like adaptive routing, quality of service guarantee, and advanced congestion control fully implemented in hardware. The slingshot switch is equipped with 64 ports, and each port is running at 200 Gbps. Slingshot also supports multiple interconnect topologies such as fat-tree and dragonfly [41]. As Cray's eighth major high-performance interconnection network technology, Slingshot is deployed on a variety of clusters like pre-exascale cluster Perlmutter [42] and exascale cluster Frontier [43], which is currently the top 1 supercomputer in the world. Slingshot is also planned to be deployed on upcoming exascale clusters like Aurora [44] and EI Capitan [45]. Slingshot is taking up 4.8% of the clusters in Top500 list as of June 2022 [19].

2.5. Aries interconnect

As Cray's third-generation interconnect architecture, Aries was introduced as part of the Cray XC system with the dragonfly topology, and it has been widely used in the HPC field [4]. A single Aries device with four NICs (Network Interface Card) and a 48-port tiled router can provide a network connection for all four nodes on a Cray XC blade. The NIC and switch in Aries are closely coupled in the dragonfly network to provide cost-effective and scalable global bandwidth. The system is configurable according to users' global bandwidth requirement, and its optical connection number can be adjusted according to the cost constraint. It also provides technologies such as adaptive routing, communication mechanisms, and synchronization mechanisms. Aries adopts the dragonfly topology and achieves congestion control by implementing Valiant's routing algorithm [46]. As of June 2022, 5% of the Top500 clusters use Aries, including Piz Daint [47] and Cori [48].

2.6. Tofu interconnect D

As one of the representatives of proprietary interconnects, Tofu [49] is an interconnect family developed by RIKEN and Fujitsu that is used for the K computer [50]. In 2018, TofuD (Tofu Interconnect D) was introduced as a new member of the Tofu family. Its main features are just as indicated by the name. The Tofu represents "torus fusion" and the letter D stands for high "density" node and "dynamic" packet slicing for "dual-rail" transfer [5]. TofuD is a proprietary torus-based [5] six-dimensional network, and it mainly supports congestion control with the family's virtual channel scheduling algorithm. Compared to previous Tofu and Tofu2 [51], TofuD has a much higher communication resource density, such as 48 cores per node. It also introduced dynamic packet slicing for the dual-rail transfer technique to solve the latency and fault tolerance issue in Tofu2. TofuD is adopted by the Fugaku [52], which was the top 1 cluster in the Top500 list at the time built in 2020 and ranked 2nd as of June 2022 [19].

2.7. Bull eXascale interconnect (BXI)

In 2015, Atos designed BXI as a new interconnect for HPC [18]. BXI is based on the scalable and reliable Portals4 [53] network programming interface and decouples computation and communication

Table 2
The summary of micro-benchmarks.

	Perftest [13]	RDMA-bench [56]	NetPerf [12]	iPerf [57–60]	qperf [61]	GPCNeT [62]
Link layer	IB, Eth (RoCE)	IB, Eth (RoCE)	Eth	Eth	IB, Eth (RoCE)	IB, Eth (RoCE), etc.
Programming Models	RDMA	RDMA	Socket	Socket	RDMA/Socket	MPI
Transport Protocols	RC/UC/UD DCT, SRD	RC/UC/UD	TCP, UDP, SCTP	TCP, UDP, SCTP	TCP, UDP, SCTP, SDP RC/UC/UD, RDS	Any protocol that can be used by MPI
Main metrics	Throughput, Average latency, Tail latency	Throughput, Average latency, Tail latency, WQE cache misses	Throughput, Average latency	Throughput, Average latency, Tail latency	Throughput, Average latency	Throughput, Average latency, Tail latency, Congestion Impact
Language	C	C	C	C	C	C
Thread Model	Single-thread	Single-thread	Single-thread	Multi-thread	Single-thread	Multi-process (MPI)
Communication pattern	P2P	P2P	P2P	P2P; multicast	P2P	P2P, collective communication
Real scenario	N	Y (w/ real applications)	N	N	N	Y
Real workload or trace	N	N	N	N	N	Y
Parameters of protocol internals	N	N	N	Y	N	N
Year of last update	2022	2018	2021	2022	2018	2021

by hardware offloading. It consists of two ASIC (Application-Specific Integrated Circuit)-based components: BXI NIC and BXI switch. The BXI NIC provides functions like OS bypass, communication offload, and reliability. Each BXI switch is equipped with forty-eight 100 Gbps ports and provides power saving and network performance monitoring functions. BXI supports multiple network topologies such as fat-tree, butterfly [54], and torus. BXI implements efficient fine-grain adaptive routing on each port basis to minimize the possibility of congestion. It also provides reliability and stability guarantees by some optimizations like deadlock-avoidance and load-balancing mechanisms. As of June 2022, Tera-1000-2 adopts BXI 1.2 and it is ranked 45th in the Top500 list [19].

2.8. Summary

We survey the above hot interconnects because of their popularity for clusters in the Top500 list. Table 1 summarizes a brief comparison of these hot interconnects. They show a huge diversity in aspects, such as bandwidth, latency, congestion control mechanism, and network topology, which motivates us to investigate their performance characteristics. Due to the lack of access to proprietary interconnects, we mainly focus on evaluating Ethernet, RoCE, IB, and Omni-Path in this paper. InfiniBand is an essential interconnect for native RDMA designs. 10/25 Gbps Ethernet networks are the majority (27.2%) of interconnects used in data center and HPC clusters. RoCE and TCP/IP can be deployed on 10/25 Gbps Ethernet [55]. We evaluate these different interconnects and show the results in Section 5.

3. Survey of micro-benchmarks

The community has designed many benchmarks to evaluate various types of interconnects. In this section, we survey six micro-benchmarks designed to measure low-level performance metrics such as latency and bandwidth. We introduce their features and discuss their pros and cons. As shown in Table 2, six publicly-available micro-benchmarks are included for comparison. The rest of this section discusses these micro-benchmarks one by one.

3.1. Perftest

Perftest [13] was developed by Mellanox and has been well maintained since 2005. The RDMA community widely uses it for latency and bandwidth performance evaluation on InfiniBand and RoCE networks. The included micro benchmarks adopt a single-thread and ping-pong communication pattern to evaluate the throughput and latency of basic RDMA operations. We can also use it to compare different transports by specifying the transport as RC, UC, UD, Raw Ethernet, and even Mellanox DCT (Dynamic Connected Transport) [63] and AWS SRD (Scalable Reliable Datagram) [64] transports that are not specified in the standard IB specification [65]. Besides the basic operations and transports, Perftest also supports the GPUDirect feature for direct inter-GPU communication through GPUDirect RDMA and the AESXTS [66] feature for data encryption and decryption scenarios using RDMA. Perftest is designed without emulating any real application traffic or traffic probability distribution. It does not allow users to choose the traffic pattern but only with a parameter to specify the message size in each test. These tests are mainly helpful for hardware or software tuning as well as for functional testing.

3.2. RDMA-bench

RDMA-bench [56] was developed by Carnegie Mellon University in 2016. Unlike Perftest, RDMA-bench is a new benchmark suite used to understand the RDMA performance in a few scenarios extracted from real applications. With the guidelines obtained from running RDMA-bench, the authors of RDMA-bench succeeded in developing a networked sequencer and a key–value store far superior to others [67].

The benchmarks in RDMA-bench can be classified into several categories: (1) application benchmarks which include HERD [68] and MICA [69] as RDMA-based key–value store systems, and DrTM-KV [70] as an RDMA-based in-memory transaction processing system; (2) micro-benchmarks which measure the throughput of outbound and inbound RDMA operations; (3) micro-benchmarks that emulate an echo server, in which users can choose different RDMA operations for the requests and responses; (4) micro-benchmarks which emulate an RPC (Remote Procedure Calls) based sequencer server using different RDMA transports and operations; (5) micro-benchmarks which emulate a complex communication scheme with configurable thread-QP ratios to the scalability evaluation; (6) micro-benchmarks which help understand low-level factors that affect RDMA performance, such as WQE cache misses of outbound READs and WRITEs, etc.

3.3. Netperf

Netperf [12] was developed by Hewlett-Packard in 2005. It is widely used to measure the performance of BSD Sockets [71] for TCP, UDP, or SCTP (Stream Control Transmission Protocol) [72] using IPv4 and IPv6, Unix domain sockets [73], and DLPI (Data Link Provider

Interface) [74]. Netperf adopts a simple client–server model without multi-threading support. The main parameters include the socket buffer size, the message size, the TCP_NODELAY option, and the test mode. There are two test modes supported in Netperf: (1) the STREAM mode, which transfers bulk data through a TCP or UDP socket; (2) the RR (Request/Response) mode, which emulates iterative requester–response transactions between the client and server. The data transmitted is synthetic. Neither different probability distributions nor real-world data trace is supported. Hewlett-Packard made a plan of version 4.x of Netperf, which aimed to support synchronized and multi-threaded benchmarking.

3.4. iPerf

iPerf [57,58] is used to evaluate the performance of TCP, UDP, and SCTP traffic with IPv4 and IPv6. It provides abundant features [57,58]: (1) iPerf adopts a multi-threaded design that can scale with the number of CPUs within a system; (2) iPerf supports tuning of various parameters that are rarely supported in Netperf, such as timing, buffers, and most importantly, the internal parameters of the protocols; (3) iPerf supports multicast tests and bidirectional tests; (4) iPerf can run on many platforms which include Linux and Windows; (5) users can get various forms of outputs in iPerf; (6) iPerf provides the libiperf library, which is an straightforward way to use and customized the functionality of iPerf.

iPerf has evolved into two incompatible active branches. One branch is iPerf2 [57] which is the newer version of the original iPerf. The other branch is iPerf3 [58] which is a redesign of the original iPerf and was now principally developed by ESnet and Lawrence Berkeley National Laboratory. Either of them contains several options and functions that are not present in the other. Generally, for TCP and UDP in Ethernet, iPerf2 and iPerf3 are about the same if running with the default configuration. However, users should check the detailed comparison in [59,60] to avoid misuse.

3.5. qperf

qperf [61] was initially developed by QLogic in 2007 and then maintained by the Linux community. qperf can measure the bandwidth and latency between two hosts using TCP, UDP, SCTP, RDMA, SDP (Sockets Direct Protocol), and RDS (Reliable Datagram Sockets). It adopts a single-threaded client–server model similar to Netperf. For RDMA, we can test the bandwidth and latency of RC, UC, and UD transports. All the operations can be measured for each transport in the tests. Compared to Perftest, qperf supports fewer transports and features from the perspective of evaluating RDMA performance. For non-RDMA protocols, the option of qperf can only change the message size. Evaluations of the internal features of the protocols cannot be done by using qperf. Even though qperf only reports average latency and fails to perform precise tail latency measurements, it is still popular as it is a handy and tool. The release of qperf is stable and the light-weight update was four years ago.

3.6. GPCNeT

The Global Performance and Congestion Network Test (GPCNeT) [62,75] was developed by Cray in 2019 to evaluate the network performance of MPI-based systems with the MPI-3.0 specification [76]. GPCNeT is compromised of two benchmarks: *network_test* and *network_load_test*.

network_test characterize the latency and bandwidth of an MPI application when it runs without network congestion. It builds the natural ring and random ring pattern such that all communication occurs over the network rather than within local groups. The communication patterns include two-sided peer-to-peer (8 bytes latency and 128K bytes bandwidth, natural and random rings), one-sided remote memory access (8 bytes latency and 128K bytes bandwidth, random ring), allreduce (8 bytes latency, random ring), and alltoall (128 bytes bandwidth, random ring).

network_load_test measure the performance of an MPI application with network congestion. This simulates the scenario when running on multi-tenant HPC networks. Each congestor has a unique random ring, and the communication patterns include Point-to-point Incast, All-to-all, One-sided RMA Incast, and One-sided RMA Broadcast. Two measurements execute in the random ring infrastructure: Point-to-point Latency measurement by sending and receiving 8 bytes messages from and to two sides, Point-to-point Bandwidth with Synchronization by sending and receiving eight 128K bytes messages from two sides.

The default settings are intended to be utilized in general production scenarios. It reports the mean and 99th percentile latencies as well as the bandwidth per rank. With congestors, it also reports the Congestion Impact metric, which is defined as the ratio of congested latency or bandwidth divided by the uncongested latency or bandwidth. The Congestion Impact metric is an indicator to study the impact of congestion across systems with different networks.

3.7. Summary

The above micro-benchmarks are surveyed because of their popularity in the community. In Table 2, we show a summary of these micro-benchmarks. Among the six micro-benchmarks, we will test the interconnects with Perftest and NetPerf in this survey. They are both widely used and well maintained since their first release. Besides, Perftest is provided by Mellanox, the most popular manufacturer of InfiniBand. Hence, we believe Perftest and NetPerf can represent defacto standard benchmarks for RDMA-based and socket-based programming models on various interconnects, respectively. We show the related results in Section 5.

4. Survey of application-level benchmarks

There are diverse types of workloads running across machines in a data center, from parallel computing to microservice, from GPU applications for deep learning workloads to Key–Value Store for big data workloads. The same issue these workloads share is that they all need efficient data communication through the interconnects. As mentioned above, different interconnects may show different characterizations on the same application. Therefore, researchers need to use benchmarks to characterize the application that runs on a specific interconnect. This section surveyed application-level benchmarks with diverse evaluation granularity for different application scenarios in data centers that involve cross-node communication via interconnects. To save space, we put detailed descriptions of these benchmarks in tables.

4.1. MPI benchmarks

MPI [15] is a message-passing standard and widely used in HPC where many processes or cores are organized to run parallel program simultaneously for acceleration. Using a benchmark to characterize MPI libraries on different interconnects can help developers understand the characteristics of interconnects and design applications in efficient ways.

We surveyed three popular MPI benchmarks. The OSU MPI Micro-Benchmarks [14] provided by Ohio State University (OSU) consist of point-to-point MPI operations, blocking/non-blocking collective MPI operations, and one-sided MPI operations. Table 3 shows the description details. The NAS Parallel Benchmarks (NPB) [77], provided by NASA, are derived from CFD (computational fluid dynamics) applications and are designed with MPI programming. Its description details are shown in Table 4. The Intel MPI Benchmarks (IMB) [78] provided by Intel perform MPI 1.0 ∼ 3.0 measurements for communication operations for a range of message sizes, which are shown in Table 5.

Table 3
The details of OSU MPI Micro-Benchmarks.

Categories	Metrics	Benchmarks	Description
Point-to-Point	Latency	osu_latency, osu_latency_mt, osu_latency_mp, osu_multi_lat	The test in ping-pong fashion with single-/multi- thread, multi-process, and multi-pair operations.
	Bandwidth	osu_bw, osu_bibw, osu_mbw_mr	The test in sending/receiving back-to-back messages with uni-/bi-directional operations.
Collective	Latency	osu_allgather, osu_allgatherv, osu_allreduce, osu_alltoall, osu_alltoallv, osu_barrier, osu_bcast, osu_gather, osu_gather, osu_reduce, osu_scatter, osu_reduce_scatter, osu_scatterv	The test of blocking MPI collective operations.
Non-Blocking Collective	Latency	osu_iallgather, osu_iallgatherv, osu_iallreduce, osu_ialltoall, osu_ialltoallv, osu_ialltoallw, osu_ibarrier, osu_ibcast, osu_igather, osu_igatherv, osu_ireduce, osu_iscatter, osu_iscatterv	The test of non-blocking collectives.
One-sided	Latency	osu_put_latency, osu_get_latency, osu_acc_latency osu_cas_latency osu_fop_latency, osu_get_acc_latency	The test for one-sided MPI Put, Get, Accumulate, Compare and Swap, Fetch and Op, and Get_accumulate.
	Bandwidth	osu_put_bw, osu_get_bw, osu_put_bibw,	The test for one-sided MPI Put and Get, in uni-/bi-direction.

Table 4
The details of NAS Parallel Benchmarks (NPB).

Benchmarks	Description
Five kernels: IS, EP, CG, MG, FT	The original eight benchmarks to mimic the computation and data movement in CFD applications.
Three pseudo applications: BT, SP, LU	
BT-MZ, SP-MZ, LU-MZ	Benchmarks with multiple levels and hybrid parallelism.
UA, BT-IO, DC, DT	Benchmarks for unstructured computation, parallel I/O, and data movement

4.2. PGAS benchmarks

PGAS (Partitioned Global Address Space) is a parallel programming model in the HPC community. PGAS is defined by communications on a shared memory space that every Processing Element (PE) can access without permission issues. Many programming languages and libraries are designed from the PGAS model, e.g., Unified Parallel C (UPC) [79] and OpenSHMEM [80]. Communication happens when the processes transfer data from the global memory or to the global memory space, including within and across a node. OSU Micro-Benchmarks also provides benchmarks on PGAS model: OpenSHMEM benchmark is shown in Table 6; UPC and UPC++ benchmarks with point-to-point (*put* and *get*) and collective communications.

4.3. RPC benchmarks

RPC is a method when a process on a machine calls procedures on other machines where the execution of the procedure happens [81]. It is a client–server interaction where data is transferred frequently over the interconnects to call (from the client) and respond (from the server) procedures. Therefore, the characteristics of the interconnect can have a direct impact on the RPC performance.

We surveyed three benchmarks for RPC applications in data centers: Apache Thrift Benchmarks (ATB), TF-gRPC-Bench, and RPC-perf. ATB is proposed in [82], which evaluates the Apache Thrift [83] based RPC performance and consists of three categories: the RPC latency evaluation benchmark, the RPC throughput evaluation benchmark, and the mixed RPC latency and throughput evaluation benchmark. Table 7 shows the details of TF-gRPC-Bench, which evaluates the communication performance between parameter server and worker process. Twitter maintains RPC-perf [84]. It is designed to evaluate the RPC's performance for caching systems regarding latency and message rate.

Table 5
The details of Intel MPI Benchmarks.

MPI-1			
Categories	Metrics	Benchmarks	Description
Single Transfer	Latency, Throughput	PingPong, PingPongSpecificSource, PingPongAnySource	Using non-blocking/blocking MPI send/recv to transfer data in ping-pong pattern.
Parallel Transfer	Latency, Message Rate, Throughput	Sendrecv, Exchange, Uniband, Biband, Multi-PingPong, Multi-PingPing, Multi-Sendrecv, Multi-Exchange, Multi-Uniband, Multi-Biband	Using blocking or non-blocking MPI send/recv to transfer data among multiple processes in uni-/bi-direction.
Collective	Latency, Throughput	Bcast/multi-Bcast, Allgather/multi-Allgather, Allgatherv/multi-Allgatherv, Alltoall/multi-Alltoall, Alltoallv/multi-Alltoallv, Scatter/multi-Scatter, Scatterv/multi-Scatterv, Gather/multi-Gather, Gatherv/multi-Gatherv, Reduce/multi-Reduce, Reduce_scatter/multi-Reduce_scatter, Allreduce/multi-Allreduce, Barrier/multi-Barrier	Using blocking MPI collective operations to transfer data among multiple processes.
IMB-P2P	Latency, Bandwidth, Message Rate	PingPong, PingPing, Unirandom, Birandom, Corandom	Using blocking or non-blocking MPI send/recv to transfer data in the shared memory.

MPI-2			
Categories	Metrics	Benchmarks	Description
IMB-EXT	Latency, Throughput	Unidir_Put, Unidir_Get, Bidir_Put, Bidir_Get, Accumulate, Window	Benchmarks in one-sided communications with uni-/bi-direction.
IMB-IO	Timing, Throughput	S_[ACTION}_indv/expl, P_[ACTION}_indv/expl /shared/priv, C_[ACTION}_indv/expl /shared, Open_Close	Input/Output (I/O) benchmarks with blocking or non-blocking I/O.

MPI-3			
Categories	Metrics	Benchmarks	Description
IMB-NBC	Timing, Overlap Percentage	Ibcast, Iallgather, Iallgatherv, Igather, Igatherv, Iscatter, Iscatterv, Ialltoall, Ialltoallv, Ireduce, Ireduce_scatter, Iallreduce, Ibarrier. Add "_pure" to measure pure time.	Using non-blocking collective operations to measure the overlap of communication and computation or the pure communication time.
IMB-RMA	Throughput	Accumulate, All_get_all, All_put_all, Bidir_get, Bidir_put, Compare_and_swap, Exchange_Get, Exchange_Put, Fetch_and_op, Get_accumulate, Get_all_local, Get_local, One_put_all, One_get_all, Put_all_local, Put_local, Truly_passive_put, Unidir_get, Unidir_put	Using one-sided communication in remote memory access (RMA) benchmarks in a uni-/bi-directional manner.

4.4. Storage benchmarks

With the development of hardware technology, much new storage hardware is produced, such as the NVMe SSD [85]. Storage systems rely on different drivers and libraries in data center, with interactions between the processors and the storage devices via interconnects. Intel SPDK [86] provides NVMe perf [87] as an NVMe SSDs benchmarking tool with minimal overhead in benchmarking. NVMe perf provides several runtime options to support the most common workload. Users

Table 6
The details of OSU OpenSHMEM Micro-Benchmarks.

Categories	Metrics	Benchmarks	Description
Point-to-Point	Latency	osu_oshm_put, osu_oshm_put_nb, osu_oshm_get, osu_oshm_get_nb	Benchmarks for PUT and GET with blocking and non-blocking routines. The users can select whether the memory allocation mode should be in global memory (non-symmetry) or heap memory (symmetry). All benchmarks use 2 PEs and the uni-directional operation. The *osu_oshm_put_overlap* provides the statistics for communication and computation overlap measurement.
	Message Rate	osu_oshm_put_mr, osu_oshm_put_mr_nb, osu_oshm_get_mr_nb, osu_oshm_put_overlap	
Collective	Latency	osu_oshm_collect, osu_oshm_fcollect, osu_oshm_broadcast, osu_oshm_reduce, osu_oshm_barrier	Benchmarks for collective operations. The users can specify the number of processes to run, and the results give the minimum, maximum, and average latency.
Atomics	Latency, Operation Rate	osu_oshm_atomics	Benchmarks for Atomics Routines. The users can select the memory allocation mode between global and heap modes.

Table 7
The details of TF-gRPC-Bench.

Categories	Description	
Point-to-Point latency	Benchmarks for measuring the latency and the bandwidth of payload transmission between a parameter server and a worker process.	The benchmarks can run on single or multiple nodes. Users can specify IP, port, warm-up period, total running period, buffer, payload generation scheme, payload serialization mode, and the number of parameter servers and workers.
Point-to-Point bandwidth		
Parameter Server Throughput	Benchmark for measuring the throughput of parameter server over gRPC.	

Table 8
The details of IOzone and Iometer.

Benchmark	Description	Metrics
IOzone	Evaluating the file system.	(Random/backwards/normal) Read/Write, Mmap, Async I/O
Iometer	Evaluating the I/O subsystem.	I/O speed, response time (avg, max), CPU utilization, error count

can configure the NVMe perf in many aspects like the workload characterizations (e.g., the percentage of Read/Write, with/without random Read/Write), the data movement protocols (e.g., PCIe, RDMA, TCP), and the execution time [87].

Besides the hardware, modern data centers also have different storage systems. Two surveyed benchmarks for storage systems are shown in Table 8. IOzone [88] is a filesystem benchmark to measure the file operations in storage systems. Iometer [89] is an I/O subsystem measurement tool for single and clustered systems. And one benchmark for EC (Erasure Coding) coder on distributed storage systems. EC-Bench [90] is an erasure coding scheme benchmark for storage architectures with description details in Table 9.

4.5. GPU applications benchmarks

GPU has been becoming incredibly popular for compute-intensive workloads in data centers and HPC clusters in recent years. Two popular deep learning frameworks, TensorFlow [91] and PyTorch [92], provide benchmarks to evaluate deep learning models, such as PerfZero [93] and TorchBench [94]. PerfZero is a benchmark framework

Table 9
The details of EC-Bench.

Categories	Description	Metrics
Encoding	A large in-memory file is split into data blocks. Each encoding operation encodes one data clock into new data and parity chunks.	Latency, Throughput, CPU Utilization, Cache Pressure
Decoding	Recover the encoded data and the parity chunks.	

Table 10
The details of PerfZero and TorchBench.

Benchmark	Platform	Metrics	Description
PerfZero	TensorFlow	wall time, CPU time, throughput, accuracy, extra information about system and environment, etc.	Evaluating and profiling for TensorFlow. Users can specify the behavior of the test, like number of iterations to run, the values to feed for each iteration, and the memory usage.
TorchBench	PyTorch	wall time, GPU time, extra information about system and environment, etc.	Evaluating and profiling for PyTorch. Users can specify the behavior of the test, like devices, modes, runs with evaluation/training.

for debugging and tracking the TensorFlow performance regression and change. TorchBench includes a collection of open-source benchmarks to evaluate models and workloads with PyTorch. More details about PerfZero and TorchBench are shown in Table 10. The PARAM benchmark [95] from Meta Platforms (Facebook formerly) can both evaluate the performance of communication components in the PyTorch deep learning framework, and evaluate the application-level workloads, like deep learning recommendation models [96,97]. DeepBench [98] produced by BaiduResearch is another benchmark for evaluating deep learning operations on different platforms. NCCL (NVIDIA Collective Communications Library) [99] and Gloo [100] provide their benchmarks on collective communication libraries, which are NCCL Tests [101] and Gloo Benchmarking [102] to evaluate the performance on collective operations.

OSU Micro-Benchmarks also provide several extensions for GPU programming models and libraries, such as CUDA [103], ROCm [104], and OpenACC [105] extensions by configuring with `--enable-cuda`, `--enable-rocm`, and `--enable-openacc` in the runtime [14].

4.6. Key-Value Store benchmarks

Key–Value Store holds a data storage model that stores associations between keys and values. Keys are primitives, and values can be primitive or complex. It is popular in the big data community and widely used in NoSQL databases in data centers because of its high efficiency and scalability. We surveyed two benchmarks for Key–Value Store. YCSB [106] (Yahoo! Cloud Serving Benchmark) is used for evaluating the performance of key–value and cloud serving stores. YCSB provides five workloads with different percentages of database operations and evaluates three metrics of performance: the latency of requests, the database performance when increasing machines, and the database performance with increasing machines while the system is running. OSU HiBD-Benchmarks [107] provide benchmarks for evaluating Memcached and HBase based Key–Value Store.

4.7. Microservice benchmarks

Microservice is a type of cloud service architecture. Unlike traditional monolithic applications, microservice consists of multiple services working together to finish a workload. Therefore, communications happen frequently among services via interconnects in data centers. We surveyed two benchmarks for microservice workloads.

Table 11
The details of DeathStarBench.

Microservice / Benchmark	Comm. Protocol	Unique Micro-services	Description	Data Storage
Social Network	RPC	36	A broadcast-style social network with posting posts, accounts and posts management, and friend recommendation functionalities.	Memcached, MongoDB
Movie Reviewing	RPC	38	A movie information management system with searching, browsing, reviewing, and rating the movies.	Memcached, MongoDB, MySQL database
E-commerce Website	REST, RPC	41	An e-commerce site for clothing with searching, browsing, ordering, and paying functionalities.	Memcached, MongoDB
Banking System	RPC	34	A secure banking system with processing payments, requesting loans, and balances credit card functionalities.	Memcached, MongoDB, Relational database
Swarm Cloud	REST, RPC	25	It runs on the cloud and edge devices with a swarm of programmable drones to perform image recognition and obstacle avoidance.	Cassandra database
Swarm Edge	REST	21		

Table 12
The details of μSuite.

Microservice / Benchmark	Description
HDSearch	HDSearch is a content-based image similarity search application service that searches the images whose vectors are near the query image's feature vector in a large image repository. The front-end sends the query to the backend, then the backend returns the results. It evaluates the performance of image search functionality.
Router	Router is a distributed Memcached service. It routes client requests to suitable Memcached servers. It evaluates the performance of GETs and SETs
Set Algebra	Set Algebra is a set intersection processing service. It performs information retrieval on posting lists by searching through millions of documents. It evaluates the performance of that searching process.
Recommend	Recommend is a recommendation system in the fields of e-commerce and behavior prediction. It predicts the users' rating for an item by using their overall preference data. It evaluates the performance of each {user, item} query.

DeathStarBench [108] is an open-source benchmark suite for microservices on cloud and edge systems, and the details are shown in Table 11. μSuite [109] can be used for evaluating the influence of OS and network on microservices, and the details are shown in Table 12.

4.8. Summary

The above-mentioned application-level benchmarks can represent a broad range of data center applications, including HPC, big data, AI, and cloud computing. In this survey, we choose MPI and PGAS based benchmarks as application examples and run them on different interconnects. MPI and PGAS based benchmarks have been designed and maintained for many years with a lot of contributed optimizations from the community. Our experience also reveals that they are easy to deploy and convenient to run. The experiment results are shown in Section 5.

5. Experiment

This section presents performance characterizations with the selected benchmarks on various hot interconnects.

Table 13
The details of the testbeds in the experiments. PADSYS and Pinnacles [110] clusters are used in Section 5.2 and Section 5.3, Bebop [111] and JLSE [112] clusters are used in Section 5.4.

Testbed (Nodes)	Interconnect (Gbps)	Intel Xeon CPU	RAM (GB)	Communication Subsystem
PADSYS (2)	InfiniBand (200)	Gold 6330	256	OFED MLNX-5.5
Pinnacles (40)	InfiniBand (100)	Gold 6330	256	RHTL 8.6 IB Driver
Bebop (36)	Omni-Path (100)	Broadwell E5-2695v4	128	Libfabric [25] v1.15.1
JLSE (13)	InfiniBand (100)	Platinum 8180M/8176	768	UCX [113,114] v1.13.0

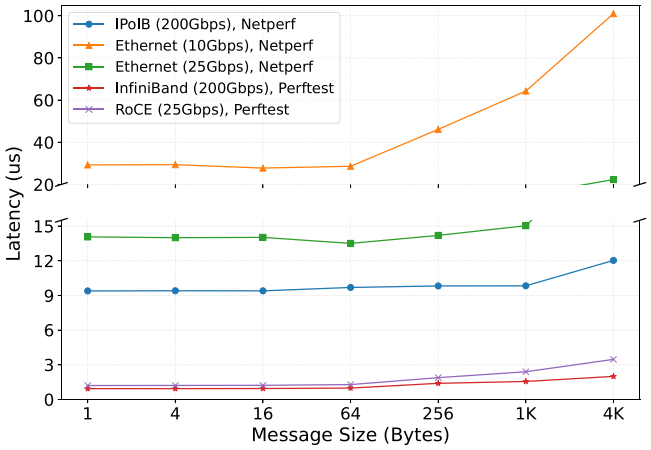


Fig. 2. The latency of *Perftest on 200 Gbps InfiniBand*, *Perftest on RoCE (25 Gbps Ethernet)*, *Netperf on 10/25 Gbps Ethernet*, and *Netperf on IPoIB (200 Gbps InfiniBand)*.

5.1. Benchmarking setup

We run benchmarks on different clusters with various interconnects and Table 13 shows the details of each cluster. We try to keep the comparisons of the experiment results as fair as possible by: (1) allocating nodes in the same rack across different experiments; (2) tuning the number of iterations (Perftest) or time duration (Netperf) of benchmark options until getting the relatively stable results.

5.2. Micro-benchmark evaluation

We organize the following experiments using two programming models, RDMA and socket, with two Perftest and Netperf micro-benchmarks on three 10/25 Gbps Ethernet and 200 Gbps InfiniBand interconnects. We discuss the experiment results in three aspects: (1) the latency comparison; (2) the bandwidth usage comparison; (3) the impact on the performance using two different InfiniBand interconnects.

5.2.1. Latency

Fig. 2 shows the latency of benchmarks based on different network. The Perftest benchmark on 200 Gbps InfiniBand, the fastest interconnect in our experiments, has the lowest latency because of the nature of kernel-bypass and high-performance protocol in RDMA. Although RoCE also supports RDMA, the hardware it uses is 25 Gbps Ethernet on our testbed which is lower than 200 Gbps InfiniBand, so the Perftest on RoCE are slower than those on InfiniBand. The Netperf [12] is a TCP benchmark running on IPoIB and 10/25 Gbps Ethernet. Due to the well-known heavy overhead of TCP [29,115], the latency numbers of these three are far slower than the native RDMA designs, and the latency becomes larger with the decrease of the network bandwidth.

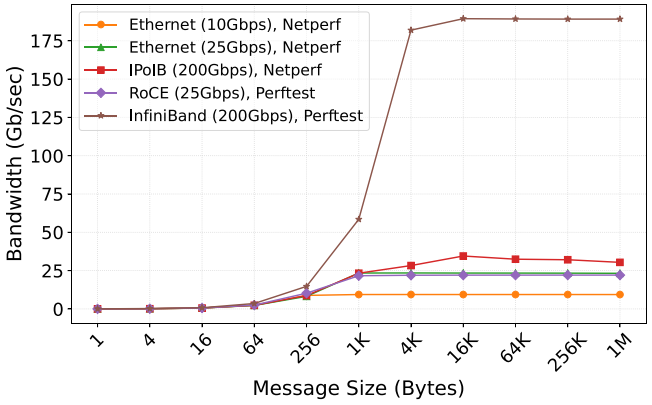


Fig. 3. The bandwidth of *Perftest on 200 Gbps InfiniBand*, *Perftest on RoCE (25 Gbps Ethernet)*, *Netperf on 10/25 Gbps Ethernet*, and *Netperf on IPoIB (200 Gbps InfiniBand)*.

5.2.2. Bandwidth

Fig. 3 shows the bandwidth comparisons using different benchmarks on different interconnects. The RDMA benchmark, Perftest, runs on 200 Gbps InfiniBand and can achieve the highest bandwidth of InfiniBand when the message size is large enough (4K bytes) because of the 4K bytes MTU setting on InfiniBand. The same benchmark running on 25 Gbps Ethernet, shown as RoCE in the figure, shows the same behavior but its MTU is 1K bytes so RDMA RoCE saturates the bandwidth earlier than the one on RDMA IB. The Netperf benchmark running on 10/25 Gbps Ethernet shows a similar behavior when the message size is larger than one MTU (1K bytes) but does not show the same on 200 Gbps InfiniBand. The reason is the TCP protocol stack overhead by deploying IPoIB on InfiniBand. We also observed the unstable results on Netperf benchmark evaluations and the reason could come from the performance fluctuation nature of TCP. Therefore, we ran the experiment five times for each one and took the average results to show in the figure.

5.2.3. InfiniBand EDR VS. HDR

200 Gbps InfiniBand (HDR) is emerging as a replacement of the widely-used 100 Gbps InfiniBand (EDR) in data center and HPC clusters. Therefore, it is high time to use benchmarks to compare the performance characteristics between EDR and HDR. In this experiment, we run the same benchmark, Perftest, on these two kinds of InfiniBand interconnects. Fig. 4 shows the bandwidth comparison and Fig. 5 shows the throughput comparison. As we expect, the saturated bandwidth (when message size is larger than 4K bytes, which equals to one MTU) of IB HDR is around two times that of IB EDR. We observe that the throughput of IB EDR is lower than the throughput of IB HDR all the time, and the IB HDR throughput numbers are 1.5X–2X times of the EDR numbers, which corresponds to the bandwidths differences between IB EDR and HDR. For *read* in 4K bytes message size, its performance is poorer than *send* and *write*. The reason is the *read* requests need to be maintained with more context overhead to wait the responses arrive [68].

5.3. MPI benchmark evaluation

This section gives the evaluation results (latency, bandwidth, and throughput) with OSU MPI Micro-Benchmarks on IB EDR and HDR.

5.3.1. Latency

The latency evaluation is shown in Fig. 6. MPI adds an extra software layer over low-level RDMA verbs. Therefore, the latency of MPI is slightly higher than that of Perftest in Section 5.2. The first observation is that running MPI on IB HDR has lower latency than on IB EDR, as expected. The second observation is that the average latency will increase with more processes usage, which scenario is closer to the real world because of the more communication overhead.

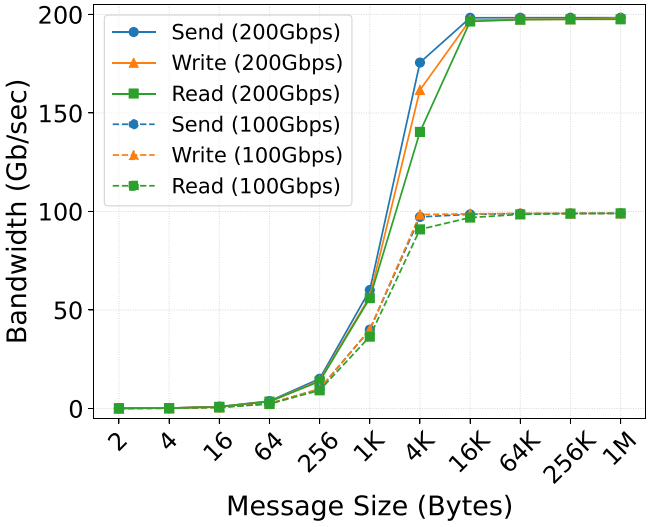


Fig. 4. Bandwidth evaluations on IB EDR and HDR.

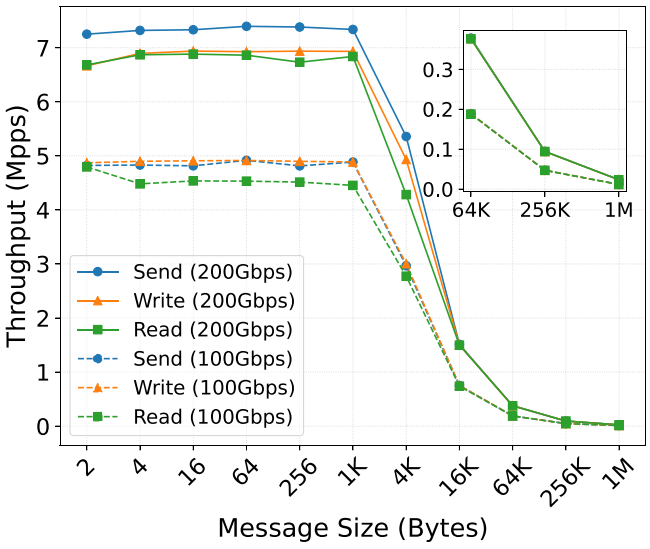


Fig. 5. Throughput evaluations on IB EDR and HDR.

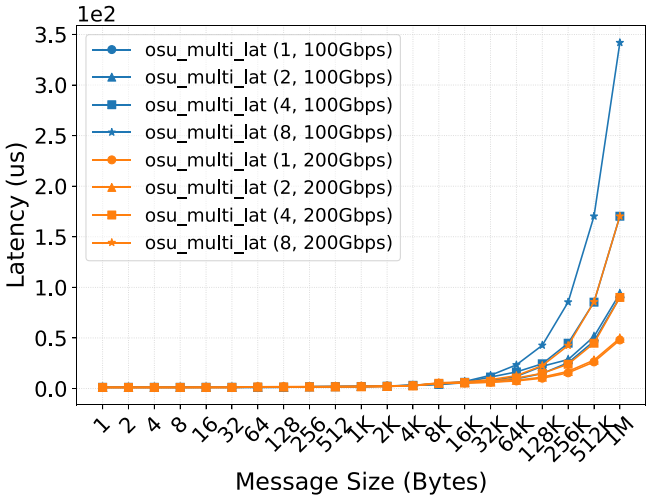
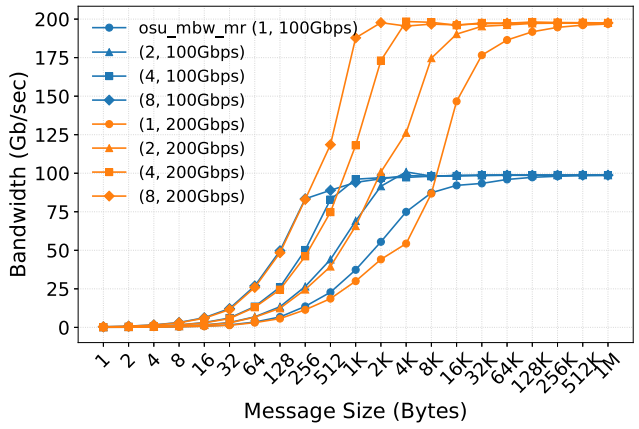
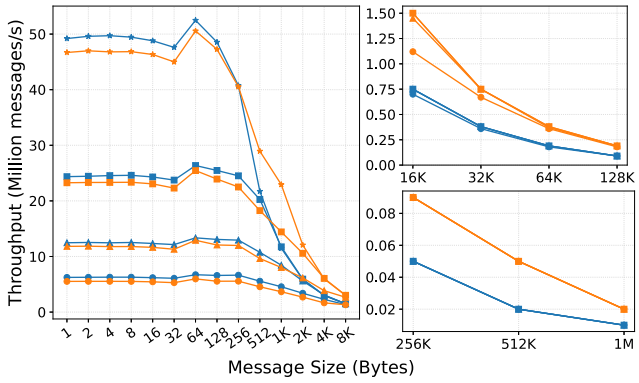


Fig. 6. OSU Micro-Benchmarks for MPI latency on multiple process pairs scenario. The first number in the bracket means the number of process pairs, and the second number is the highest bandwidth of InfiniBand.



(a) Bandwidth comparison on multiple processes scenario. The first number in the bracket means the number of processes, and the second number is the highest bandwidth of InfiniBand.



(b) Throughput on multiple processes scenario. (The legend is same as Figure 7a)

Fig. 7. The OSU Micro-Benchmarks for MPI bandwidth and throughput on multiple processes scenario.

5.3.2. Bandwidth

We also see the same trend as in Section 5.2.2 when the MPI benchmark runs on different IB interconnects in Fig. 7(a): The saturated bandwidth on IB HDR is two times the saturated bandwidth on IB EDR. The more processes are used, the earlier the bandwidth will be saturated.

5.3.3. Throughput

Although in Section 5.2.3 we can see that when the message size is the same, the throughput of Perftest on IB EDR is lower than that on IB HDR, we do not get the exact same behavior on the throughput of OSU Micro-Benchmarks for MPI which is shown in Fig. 7(b). The reason comes from different aspects. When the message size is small, MPI cannot saturate the bandwidth, so the throughput is almost the same at that stage. When the message size becomes larger, the throughput decreases rapidly and starts to saturate the bandwidth. We can observe that the throughput of IB HDR is around two times that of IB EDR, corresponding to the bandwidth ratio between two InfiniBand interconnects.

5.4. PGAS benchmark evaluation

We use OSU Micro-Benchmarks for OpenSHMEM to characterize the performance of running OpenSHMEM benchmarks on different interconnects. We use Sandia-OpenSHMEM (SOS) [116] because SOS

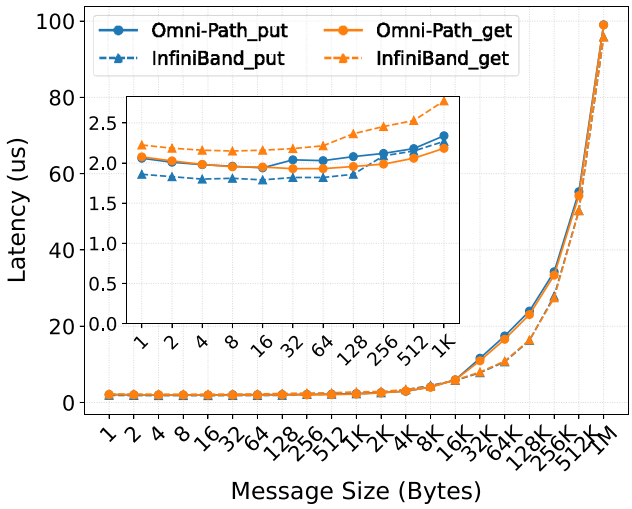


Fig. 8. OSU Micro-Benchmarks for OpenSHMEM for point-to-point communication on Omni-Path and InfiniBand.

is one of the native OpenSHMEM implementations. To evaluate how interconnect influences the OpenSHMEM performance, we evaluate the latency of point-to-point communication on 2 nodes (1 PE per node) and collective communication on 8 nodes (1 PE per node) on two different interconnects: Omni-Path Fabric and InfiniBand with the same 100 Gbps bandwidth. Fig. 8 shows the latency performance comparison that are divided into two parts, point-to-point operations: *put* and *get*, and Fig. 9 shows the comparison of collective operations: *broadcast* (one-to-all) and *alltoall* (all-to-all).

For the point-to-point communication in Fig. 8, the latency results on Omni-Path and InfiniBand are comparable in most cases. The latency of *get* operation on Omni-Path is slightly better than that on IB with medium message size (~1K bytes). When it goes to large messages (16–512K bytes), the latency of IB is better than that of Omni-Path for both *put* and *get* operations. Although SOS has some specific optimizations on Omni-Path, we do not observe the corresponding optimizations compared with IB for *put* and *get*, which could attribute to the heterogeneous hardware configurations, like the cache size or CPU, on two clusters.

The collective communication performance is shown in Fig. 9. InfiniBand shows a lower latency number than Omni-Path in most cases, except for *alltoall* operation in small message size. Especially, the latency number of *alltoall* operation on Omni-Path is much slower than the one on InfiniBand. In addition to the heterogeneous hardware configurations, the different network frameworks (Libfabric on Omni-Path and UCX on InfiniBand) could also be why the performance is different.

6. Discussion

This section summarizes some improvable aspects of existing benchmarks as we have observed after showing the example experiments and performance characterization results. (1) *It is not always easy to get stable numbers.* As we mentioned earlier, many benchmarks are not stable, and we may need to tune many parameters carefully or take the average of multiple rounds of running to get stable numbers. (2) *Not many benchmarks provide reference numbers.* Some benchmarks do not give reference numbers so that the users cannot evaluate whether their results are reasonable or not. Hence, we encourage our community to publish more reference numbers with the surveyed benchmarks in this paper on various interconnects as guidance. (3) *Some benchmarks lack clear instructions or specifications.* Some benchmarks assume that users are experts and do not provide clear instructions or specifications

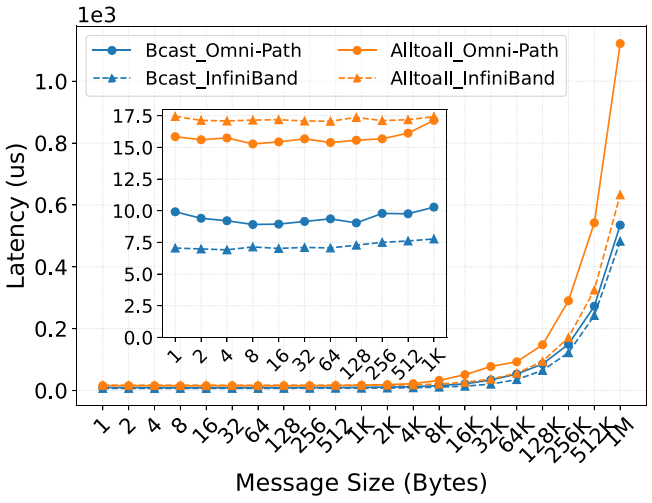


Fig. 9. OSU Micro-Benchmarks for OpenSHMEM with Broadcast and Alltoall on Omni-Path and InfiniBand.

about benchmark installation, configuration, and usage information. (4) *Not many benchmarks provide detailed warnings.* We find that the proper warning messages are helpful for benchmarking. For example, Perftest can warn the users when 'CPU frequency is not max'. (5) *Benchmarks usually use different ways to calculate and present numbers.* Some benchmarks may use number of iterations to characterize performance, while some use time duration. Some benchmarks present the best performance numbers, while some benchmarks give the average performance numbers. Therefore, users need to compare the numbers across benchmarks carefully.

7. Related work

Besides the related survey studies and benchmarks discussed in Section 1, 3, and 4, this section summarizes more benchmarking studies.

Benchmarking distributed storage systems: The distributed storage benchmarks evaluate how the storage system serves requests for reading and writing files and objects. For example, SKB [117] supports performance benchmarking of 43 distributed storage systems. The Cloud Object Storage Benchmark [118] is for benchmarking cloud object storage services. Acquaviva et al. [119] developed a benchmark to evaluate different Cloud Distributed File Systems.

Benchmarking big data systems: Many benchmarks are proposed to evaluate the big data systems with the big data boom. HiBench [120] and MRBench [121] are designed for evaluating MapReduce systems. TextBenDS [122] is applied to evaluate the performance of Hive, Spark, and MongoDB on a textual corpus. The TPC [123] organization designed benchmark standards what were data-centric benchmark and disseminated verifiable data to the industry. Wang et al. [124] discussed the challenges of using the widely-used benchmarks (TPC-C and YCSB) for systems evaluation. DCQCN [125] and DSCP-BASEDPFC [126] introduce how to benchmark and monitor the RDMA traffic on data centers with RoCEv2 networks.

Benchmarking AI systems: Both AIBench [127–129] and MLPerf [130–132] cover a broad diversity of scenarios to evaluate the AI systems. DataPerf [133] benchmarks the datasets in machine learning and the algorithms in processing these datasets. HPC AI500 [134] is a benchmark suite to evaluate HPC systems that run real-world workloads.

Benchmarking computing systems: SPEC (Standard Performance Evaluation Corporation) [135] designed standardized benchmarks and tools to evaluate performance and energy efficiency for computing systems. PARSEC (Princeton Application Repository for Shared-Memory Computers) [136] benchmarks the workloads and shared-memory programs for chip-multiprocessors and contains thirteen programs in different areas.

To the best of our knowledge, we are the first to extensively survey the benchmarks for hot interconnects.

8. Conclusion

This paper presents an extensive survey on hot interconnects in modern data centers and HPC clusters and associated benchmarks to help the community understand these advanced interconnects better. After introducing some representative modern interconnects and their features, we survey some commonly used micro-benchmarks and application-level benchmarks that can be used on these interconnects to measure their performance. Based on the micro-/application-level benchmarks survey, we conduct experiments on some kinds of real interconnects with the corresponding benchmarks, illustrate performance characteristics of these interconnects, and provide our interpretation of the experiment results. Considering the continuous evolution of data center interconnects and benchmarks in the future, we also discuss existing benchmarks' improvable aspects and our insights for future benchmark design and development.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the NSF research grant CCF #2132049. Part of this research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration, and by the U.S. Department of Energy, Office of Science, under Contract DE-AC02-06CH11357. We gratefully acknowledge the computing resources provided on Bebop, an HPC cluster operated by the Laboratory Computing Resource Center at Argonne National Laboratory, USA. Part of this research was conducted using Pinnacles (NSF MRI, #2019144) at the Cyberinfrastructure and Research Technologies (CIRT) at University of California, Merced.

References

[1] D.K. Panda, X. Lu, D. Shankar, High-Performance Big Data Computing, The MIT Press, 2022.
[2] Mellanox, Introducing 200G HDR InfiniBand Solutions, 2022, https://network.nvidia.com/sites/default/files/pdf/whitepapers/WP_Introducing_200G_HDR_InfiniBand_Solutions.pdf.
[3] Slingshot, 2022, https://www.hpe.com/us/en/compute/hpc/slingshot-interconnect.html.
[4] B. Alverson, E. Froese, L. Kaplan, D. Roweth, Cray XC Series Network, 2012, Cray Inc., White Paper WP-Aries01-1112.
[5] Y. Ajima, T. Kawashima, T. Okamoto, N. Shida, K. Hirai, T. Shimizu, S. Hiramoto, Y. Ikeda, T. Yoshikawa, K. Uchida, et al., The Tofu Interconnect D, in: 2018 IEEE International Conference on Cluster Computing, CLUSTER, IEEE, 2018, pp. 646–654.
[6] 100 Gigabit Ethernet, 2022, https://en.wikipedia.org/wiki/100_Gigabit_Ethernet.
[7] Terabit Ethernet, 2022, https://en.wikipedia.org/wiki/Terabit_Ethernet.
[8] R. Han, X. Lu, J. Xu, On Big Data Benchmarking, in: Workshop on Big Data Benchmarks, Performance Optimization, and Emerging Hardware, Springer, 2014, pp. 3–18.
[9] Q. Zhang, L. Zha, J. Lin, D. Tu, M. Li, F. Liang, R. Wu, X. Lu, A Survey on Deep Learning Benchmarks: Do We Still Need New Ones? in: International Symposium on Benchmarking, Measuring and Optimization, Springer, 2018, pp. 36–49.
[10] X. Zhou, X. Peng, T. Xie, J. Sun, C. Xu, C. Ji, W. Zhao, Poster: Benchmarking Microservice Systems for Software Engineering Research, in: 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion), IEEE, 2018, pp. 323–324.

[11] W. Gao, J. Zhan, L. Wang, C. Luo, D. Zheng, X. Wen, R. Ren, C. Zheng, X. He, H. Ye, et al., BigDataBench: A Scalable and Unified Big Data and AI Benchmark Suite, 2018, arXiv preprint arXiv:1802.08254.

[12] Netperf, 2022, https://github.com/HewlettPackard/netperf.

[13] Open Fabrics Enterprise Distribution (OFED) Performance Tests, 2022, https://github.com/linux-rdma/perftest.

[14] D.K. Panda, H. Subramoni, C.-H. Chu, M. Bayatpour, The MVAPICH project: Transforming Research into High-performance MPI Library for HPC Community, J. Comput. Sci. 52 (2021) 101208, http://dx.doi.org/10.1016/j.jocs.2020.101208, URL https://www.sciencedirect.com/science/article/pii/S1877750320305093. Case Studies in Translational Computer Science.

[15] M.P.I. Forum, MPI: A Message-Passing Interface Standard Version 4.0, 2021, URL https://www.mpi-forum.org/docs/mpi-4.0/mpi40-report.pdf.

[16] G. Almasi, PGAS (Partitioned Global Address Space) Languages, in: D. Padua (Ed.), Encyclopedia of Parallel Computing, Springer US, Boston, MA, 2011, pp. 1539–1545, http://dx.doi.org/10.1007/978-0-387-09766-4_210.

[17] OmniPath, 2022, https://www.cornelisnetworks.com/products/.

[18] S. Derradji, T. Palfer-Sollier, J.-P. Panziera, A. Poudes, F.W. Atos, The BXI Interconnect Architecture, in: 2015 IEEE 23rd Annual Symposium on High-Performance Interconnects, IEEE, 2015, pp. 18–25.

[19] TOP 500 Supercomputer Sites, 2022, http://www.top500.org.

[20] I.T. Association, et al., Supplement to InfiniBand Architecture Specification, Release 1 (2) (2010) 1.

[21] IEEE std 802.3x-1997 and IEEE std 802.3y-1997 (supplement to ISO/IEC 8802-3: 1996, in: ANSI/IEEE Std 802.3, 1996 ed., IEEE Standards for Local and Metropolitan Area Networks: Supplements to Carrier Sense Multiple Access with Collision Detection (CSMA/CD).

[22] S. Floyd, TCP and Explicit Congestion Notification, ACM SIGCOMM Comput. Commun. Rev. 24 (5) (1994) 8–23.

[23] M. Alizadeh, B. Atikoglu, A. Kabbani, A. Lakshmikantha, R. Pan, B. Prabhakar, M. Seaman, Data Center Transport Mechanisms: Congestion Control Theory and IEEE Standardization, in: 2008 46th Annual Allerton Conference on Communication, Control, and Computing, IEEE, 2008, pp. 1270–1277.

[24] Elastic Fabric Adapter - Amazon Elastic Compute Cloud, 2022, https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/efa.html.

[25] OpenFabrics Interfaces Working Group, Libfabric OpenFabrics, 2022, URL https://ofiwg.github.io/libfabric/.

[26] Summit, 2022, https://www.olcf.ornl.gov/summit/.

[27] Sierra, 2022, https://hpc.llnl.gov/hardware/compute-platforms/sierra.

[28] InfiniBand Trade Association, InfiniBand Architecture Specification : Release 1.2.1, 2007, https://www.afs.enea.it/asantoro/V1r1_2_1.Release_12062007.pdf.

[29] X. Lu, D. Shankar, S. Gugnani, H. Subramoni, 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 Computer Society, Los Alamitos, CA, USA, 2016, pp. 310–317, http://dx.doi.org/10.1109/CloudCom.2016.0057, URL https://doi.ieeecomputersociety.org/10.1109/CloudCom.2016.0057.

[30] M. Al-Fares, A. Loukissas, A. Vahdat, A Scalable, Commodity Data Center Network Architecture, ACM SIGCOMM Comput. Commun. Rev. 38 (4) (2008) 63–74.

[31] A. Shpiner, Z. Haramaty, S. Eliad, V. Zdornov, B. Gafni, E. Zahavi, Dragonfly+: Low Cost Topology for Scaling Datacenters, in: 2017 IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB), IEEE, 2017, pp. 1–8.

[32] J. Jiang, R. Jain, C. So-In, An Explicit Rate Control Framework for Lossless Ethernet Operation, in: 2008 IEEE International Conference on Communications, IEEE, 2008, pp. 5914–5918.

[33] P. Newman, Backward Explicit Congestion Notification for ATM Local Area Networks, in: Proceedings of GLOBECOM'93. IEEE Global Telecommunications Conference, IEEE, 1993, pp. 719–723.

[34] GPUDirect RDMA, 2022, https://docs.nvidia.com/cuda/gpudirect-rdma/index.html.

[35] A. Sodani, Knights Landing (KNL): 2nd Generation Intel® Xeon Phi Processor, in: 2015 IEEE Hot Chips 27 Symposium, HCS, IEEE, 2015, pp. 1–24.

[36] Cornelis Networks, 2022, https://www.cornelisnetworks.com/.

[37] A. Tekin, A. Tuncer Durak, C. Piechurski, D. Kaliszan, F. Aylin Sungur, F. Robertsén, P. Gschwandtner, State-of-The-Art and Trends for Computing and Interconnect Network Solutions for HPC and AI, 2021, Partnership for Advanced Computing in Europe, Available Online At www.Praceri.Eu.

[38] QSFP28 Adapter, 2022, https://www.ibm.com/docs/en/power9?topic=ad-pcie3-2-port-100-gbe-nic-roce-qsfp28-adapter-fc-ec3l-ec3m-ccin-2cec.

[39] NVIDIA Mellanox InfiniBand NDR 400G Architecture, 2022, https://applieddatasystems.com/wp-content/uploads/2020/11/br-ndr-architecture-brochure.pdf.

[40] About Fugaku, 2022, https://www.r-ccs.riken.jp/en/fugaku/about/.

[41] J. Kim, W.J. Dally, S. Scott, D. Abts, Technology-driven, Highly-scalable Dragonfly Topology, in: 2008 International Symposium on Computer Architecture, IEEE, 2008, pp. 77–88.

[42] Perlmutter, 2022, https://www.nersc.gov/systems/perlmutter/.

[43] Frontier, 2022, https://www.olcf.ornl.gov/frontier/.

[44] Aurora, 2022, https://www.alcf.anl.gov/aurora.

[45] EI Capitan, 2022, https://www.hpe.com/us/en/compute/hpc/cray/doe-el-capitan-press-release.html.

[46] L.G. Valiant, A Scheme for Fast Parallel Communication, SIAM J. Comput. 11 (2) (1982) 350–361.

[47] Piz Daint, 2022, https://www.cscs.ch/computers/piz-daint/.

[48] Cori, 2022, https://docs.nersc.gov/systems/cori/.

[49] Y. Ajima, S. Sumimoto, T. Shimizu, Tofu: A 6D Mesh/Torus Interconnect for Exascale Computers, Computer 42 (11) (2009) 36–40.

[50] Y. Ajima, T. Inoue, S. Hiramoto, T. Shimizu, Tofu: Interconnect for the K Computer, Fujitsu Sci. Tech. J. 48 (3) (2012) 280–285.

[51] Y. Ajima, T. Inoue, S. Hiramoto, S. Uno, S. Sumimoto, K. Miura, N. Shida, T. Kawashima, T. Okamoto, O. Moriyama, et al., Tofu Interconnect 2: System-on-chip Integration of High-performance Interconnect, in: International Supercomputing Conference, Springer, 2014, pp. 498–507.

[52] Fugaku, 2022, https://www.fujitsu.com/global/about/innovation/fugaku/.

[53] B.W. Barrett, R.B. Brightwell, R.E. Grant, K.S. Hemmert, K.T. Pedretti, K.B. Wheeler, K.D. Underwood, R. Riesen, A.B. Maccabe, T. Hudson, The Portals 4.0. 2 Networking Programming Interface, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States); Intel Corp ..., 2014.

[54] D.-G. Sun, Z.-H. Weng, Butterfly Interconnection Implementation for an n-bit Pparallel Ripple Carry Full Adder, Appl. Opt. 30 (14) (1991) 1781–1785.

[55] A. Kashyap, X. Lu, NVMe-OAF: Towards adaptive NVMe-of for IO-intensive workloads on HPC cloud, in: Proceedings of the 31st International Symposium on High-Performance Parallel and Distributed Computing, HPDC '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 56–70, http://dx.doi.org/10.1145/3502181.3531476.

[56] RDMA-bench: A Framework to Understand RDMA Performance, 2022, https://github.com/efficient/rdma_bench.

[57] iPerf2 - A Means to Measure Network Responsiveness and Throughput, 2022, https://sourceforge.net/projects/iperf2/.

[58] iPerf3 - The Ultimate Speed Test Tool for TCP, UDP and SCTP, 2022, https://iperf.fr/.

[59] Comparison Table of iPerf2 and iPerf3, 2022, https://iperf2.sourceforge.io/IperfCompare.html.

[60] Change between iPerf 2.0, iPerf 3.0 and iPerf 3.1, 2022, https://iperf.fr/iperf-doc.php.

[61] qperf: Measure RDMA and IP performance, 2022, https://github.com/linux-rdma/qperf.

[62] GPCNet, 2022, https://github.com/netbench/GPCNET.

[63] H. Subramoni, K. Hamidouche, A. Venkatesh, S. Chakraborty, D.K. Panda, Designing MPI Library with Dynamic Connected Transport (DCT) of InfiniBand: Early Experiences, in: International Supercomputing Conference, Springer, 2014, pp. 278–295.

[64] L. Shalev, H. Ayoub, N. Bshara, E. Sabbag, A Cloud-Optimized Transport Protocol for Elastic and Scalable HPC, IEEE Micro 40 (6) (2020) 67–73.

[65] Transport Modes - RDMA Aware Programming User Manual v1.7, 2022, https://docs.nvidia.com/networking/display/RDMAAwareProgrammingv17/Transport+Modes#.

[66] IEEE Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices, IEEE Std 1619-2007 (2008) 1–40, http://dx.doi.org/10.1109/IEEESTD.2008.4493450.

[67] A. Kalia, M. Kaminsky, D.G. Andersen, Design Guidelines for High Performance RDMA systems, in: 2016 USENIX Annual Technical Conference (USENIX ATC 16), 2016, pp. 437–450.

[68] A. Kalia, M. Kaminsky, D.G. Andersen, Using RDMA Efficiently for Key-Value Services, in: Proceedings of the 2014 ACM Conference on SIGCOMM, SIGCOMM '14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 295–306, http://dx.doi.org/10.1145/2619239.2626299.

[69] H. Lim, D. Han, D.G. Andersen, M. Kaminsky, MICA: A Holistic Approach to Fast In-Memory Key-Value Storage, in: 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), 2014, pp. 429–444.

[70] X. Wei, J. Shi, Y. Chen, R. Chen, H. Chen, Fast In-memory Transaction Processing Using RDMA and HTM, in: Proceedings of the 25th Symposium on Operating Systems Principles, 2015, pp. 87–104.

[71] Wikipedia contributors, Berkeley Sockets — Wikipedia, the free encyclopedia, 2022.

[72] Wikipedia contributors, Stream Control Transmission Protocol — Wikipedia, the free encyclopedia, 2022, https://en.wikipedia.org/w/index.php?title=Stream_Control_Transmission_Protocol&oldid=1091520392 (Online; Accessed 12 September 2022).

[73] W. contributors, Unix Domain Socket — Wikipedia, the free encyclopedia, 2022, https://en.wikipedia.org/w/index.php?title=Unix_domain_socket&oldid=1100609125 (Online; Accessed 12 September 2022).

[74] The Open Group, Data Link Provider Interface (DLPI), 2022, https://pubs.opengroup.org/onlinepubs/009638599/toc.htm (Online; accessed 12 September 2022).

[75] S. Chunduri, T. Groves, P. Mendygral, B. Austin, J. Balma, K. Kandalla, K. Kumaran, G. Lockwood, S. Parker, S. Warren, N. Wichmann, N. Wright, GPCNeT: Designing a Benchmark Suite for Inducing and Measuring Contention in HPC Networks, SC '19, Association for Computing Machinery, New York, NY, USA, 2019, http://dx.doi.org/10.1145/3295500.3356215.

[76] M.P.I. Forum, MPI: A Message-Passing Interface Standard Version 3.0, 2012, URL https://www.mpi-forum.org/docs/mpi-3.0/mpi30-report-book.pdf.
[77] NAS Parallel Benchmarks, 2022, https://www.nas.nasa.gov/software/npb.html.
[78] Intel® MPI Benchmarks, 2022, https://www.intel.com/content/www/us/en/developer/articles/technical/intel-mpi-benchmarks.html.
[79] U. Consortium, D. Bonachea, G. Funck, UPC Language and Library Specifications, Version 1.3, 2013, http://dx.doi.org/10.2172/1134233, URL https://www.osti.gov/biblio/1134233.
[80] OpenSHMEM Application Programming Interface, v1. 5 Final, 2022.
[81] M. Van Steen, A.S. Tanenbaum, Distributed Systems, Maarten van Steen Leiden, The Netherlands, 2017.
[82] T. Li, H. Shi, X. Lu, HatRPC: Hint-Accelerated Thrift RPC over RDMA, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '21, Association for Computing Machinery, New York, NY, USA, 2021, http://dx.doi.org/10.1145/3458817.3476191.
[83] Apache Thrift, 2022, https://thrift.apache.org/.
[84] rpc-perf, 2022, https://github.com/twitter/rpc-perf.
[85] Wikipedia contributors, NVM Express — Wikipedia, the free encyclopedia, 2022, https://en.wikipedia.org/w/index.php?title=NVM_Express&oldid=1105967834 (Online; accessed 13-September-2022).
[86] Z. Yang, J.R. Harris, B. Walker, D. Verkamp, C. Liu, C. Chang, G. Cao, J. Stern, V. Verma, L.E. Paul, SPDK: A Development Kit to Build High Performance Storage Applications, in: 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, 2017, pp. 154–161.
[87] SPDK: NVMe Driver, 2022, https://spdk.io/doc/nvme.html.
[88] IOzone Filesystem Benchmark, 2022, https://www.iozone.org/.
[89] Iometer project, 2022, http://www.iometer.org/.
[90] H. Shi, X. Lu, D.K. Panda, EC-Bench: Benchmarking Onload and Offload Erasure Coders on Modern Hardware Architectures, in: C. Zheng, J. Zhan (Eds.), Benchmarking, Measuring, and Optimizing, Springer International Publishing, Cham, 2019, pp. 215–230.
[91] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., TensorFlow: A System for Large-Scale Machine Learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 2016, pp. 265–283.
[92] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., PyTorch: An Imperative Style, High-Performance Deep Learning Library, Adv. Neural Inf. Process. Syst. 32 (2019).
[93] PerfZero, 2022, https://github.com/tensorflow/benchmarks/tree/master/perfzero.
[94] TorchBench, 2022, https://github.com/pytorch/benchmark.
[95] Meta, PARAM, 2022, https://github.com/facebookresearch/param.
[96] M. Naumov, D. Mudigere, H.M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C. Wu, A.G. Azzolini, D. Dzhulgakov, A. Mallevich, I. Cherniavskii, Y. Lu, R. Krishnamoorthi, A. Yu, V. Kondratenko, S. Pereira, X. Chen, W. Chen, V. Rao, B. Jia, L. Xiong, M. Smelyanskiy, Deep Learning Recommendation Model for Personalization and Recommendation Systems, 2019, CoRR arXiv:1906.00091.
[97] Deep Learning Recommendation Model for Personalization and Recommendation Systems, 2022, https://github.com/facebookresearch/dlrm.
[98] BaiduResearch, DeepBench, 2022, https://github.com/baidu-research/DeepBench.
[99] S. Jeaugey, Nccl 2.0, in: GPU Technology Conference, GTC, vol. 2, 2017.
[100] Gloo, 2022, https://github.com/facebookincubator/gloo.
[101] NCCL Tests, 2022, https://github.com/NVIDIA/nccl-tests.
[102] Gloo, 2022, https://github.com/facebookincubator/gloo.
[103] D. Kirk, et al., NVIDIA CUDA Software and GPU Parallel Computing Architecture, in: ISMM, vol. 7, 2007, pp. 103–104.
[104] ROCm, 2022, https://www.amd.com/en/graphics/servers-solutions-rocm.
[105] OpenACC, 2022, https://www.openacc.org/.
[106] B.F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, R. Sears, Benchmarking Cloud Serving Systems with YCSB, in: Proceedings of the 1st ACM Symposium on Cloud Computing, Association for Computing Machinery, New York, NY, USA, 2010, pp. 143–154, http://dx.doi.org/10.1145/1807128.1807152.
[107] The High-Performance Big Data Project, 2022, https://hibd.cse.ohio-state.edu/.
[108] Y. Gan, Y. Zhang, D. Cheng, A. Shetty, P. Rathi, N. Katarki, A. Bruno, J. Hu, B. Ritchken, B. Jackson, K. Hu, M. Pancholi, Y. He, B. Clancy, C. Colen, F. Wen, C. Leung, S. Wang, L. Zaruvinsky, M. Espinosa, R. Lin, Z. Liu, J. Padilla, C. Delimitrou, An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud & Edge Systems, in: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, Association for Computing Machinery, New York, NY, USA, 2019, pp. 3–18, http://dx.doi.org/10.1145/3297858.3304013.
[109] A. Sriraman, T.F. Wenisch, μSuite: A benchmark suite for microservices, in: 2018 IEEE International Symposium on Workload Characterization, 2018, pp. 1–12, http://dx.doi.org/10.1109/IISWC.2018.8573515.
[110] Pinnacles cluster, 2022, https://ucmerced.github.io/hpc_docs/#/Pinnacles.
[111] Bebop, 2022, https://www.lcrc.anl.gov/systems/resources/bebop/.
[112] Joint Laboratory for System Evaluation (JLSE), 2022, https://www.jlse.anl.gov/.
[113] The Unified Communication X Library, 2022, http://www.openucx.org.
[114] P. Shamis, M.G. Venkata, M.G. Lopez, M.B. Baker, O. Hernandez, Y. Itigin, M. Dubman, G. Shainer, R.L. Graham, L. Liss, et al., UCX: An Open Source Framework for HPC Network APIs and Beyond, in: 2015 IEEE 23rd Annual Symposium on High-Performance Interconnects, IEEE, 2015, pp. 40–43.
[115] X. Lu, D. Shankar, S. Gugnani, D.K. Panda, High-Performance Design of Apache Spark with RDMA and Its Benefits on Various Workloads, in: 2016 IEEE International Conference on Big Data (Big Data), 2016, pp. 253–262, http://dx.doi.org/10.1109/BigData.2016.7840611.
[116] Sandia-OpenSHMEM/SOS, 2022, https://github.com/Sandia-OpenSHMEM/SOS.
[117] K. Munegowda, N. Sanjay Kumar, Design and Implementation of Storage Benchmark Kit, in: Emerging Research in Computing, Information, Communication and Applications, Springer, 2022, pp. 45–62.
[118] Q. Zheng, H. Chen, Y. Wang, J. Duan, Z. Huang, COSBench: A Benchmark Tool for Cloud Object Storage Services, in: 2012 IEEE Fifth International Conference on Cloud Computing, IEEE, 2012, pp. 998–999.
[119] L. Acquaviva, P. Bellavista, A. Corradi, L. Foschini, L. Gioia, P.C.M. Picone, Cloud Distributed File Systems: A Benchmark of HDFS, Ceph, GlusterFS, and XtremeFS, in: 2018 IEEE Global Communications Conference, IEEE, 2018, pp. 1–6.
[120] S. Huang, J. Huang, J. Dai, T. Xie, B. Huang, The HiBench Benchmark Suite: Characterization of the MapReduce-based Data Analysis, in: 2010 IEEE 26th International Conference on Data Engineering Workshops, ICDEW 2010, IEEE, 2010, pp. 41–51.
[121] K. Kim, K. Jeon, H. Han, S.-g. Kim, H. Jung, H.Y. Yeom, MRBench: A Benchmark for Mapreduce Framework, in: 2008 14th IEEE International Conference on Parallel and Distributed Systems, IEEE, 2008, pp. 11–18.
[122] C.-O. Truică, E.-S. Apostol, J. Darmont, I. Assent, TextBenDS: a Generic Textual Data Benchmark for Distributed Systems, Inform. Syst. Front. 23 (1) (2021) 81–100.
[123] The Transaction Processing Performance Council (TPC), 2022, https://www.tpc.org/.
[124] Y. Wang, M. Yu, Y. Hui, F. Zhou, Y. Huang, R. Zhu, X. Ren, T. Li, X. Lu, A Study of Database Performance Sensitivity to Experiment Settings, Proc. VLDB Endow. 15 (7) (2022) 1439–1452, http://dx.doi.org/10.14778/3523210.3523221.
[125] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M.H. Yahia, M. Zhang, Congestion Control for Large-Scale RDMA Deployments, in: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, Association for Computing Machinery, New York, NY, USA, 2015, pp. 523–536, http://dx.doi.org/10.1145/2785956.2787484.
[126] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, M. Lipshteyn, RDMA over Commodity Ethernet at Scale, in: Proceedings of the 2016 ACM SIGCOMM Conference, Association for Computing Machinery, New York, NY, USA, 2016, pp. 202–215, http://dx.doi.org/10.1145/2934872.2934908.
[127] AIBench, 2022, https://www.benchcouncil.org/aibench/.
[128] W. Gao, F. Tang, J. Zhan, X. Wen, L. Wang, Z. Cao, C. Lan, C. Luo, X. Liu, Z. Jiang, Aibench Scenario: Scenario-distilling AI Benchmarking, in: 2021 30th International Conference on Parallel Architectures and Compilation Techniques, IEEE, 2021, pp. 142–158.
[129] F. Tang, W. Gao, J. Zhan, C. Lan, X. Wen, L. Wang, C. Luo, Z. Cao, X. Xiong, Z. Jiang, et al., AIBench Training: Balanced Industry-standard AI Training Benchmarking, in: 2021 IEEE International Symposium on Performance Analysis of Systems and Software, IEEE, 2021, pp. 24–35.
[130] P. Mattson, V.J. Reddi, C. Cheng, C. Coleman, G. Diamos, D. Kanter, P. Micikevicius, D. Patterson, G. Schmuelling, H. Tang, et al., MLPerf: An Industry Standard Benchmark Suite for Machine Learning Performance, IEEE Micro 40 (2) (2020) 8–16.
[131] P. Mattson, C. Cheng, G. Diamos, C. Coleman, P. Micikevicius, D. Patterson, H. Tang, G.-Y. Wei, P. Bailis, V. Bittorf, et al., MLPerf Training Benchmark, Proc. Mach. Learn. Syst. 2 (2020) 336–349.
[132] C. Banbury, V.J. Reddi, P. Torelli, J. Holleman, N. Jeffries, C. Kiraly, P. Montino, D. Kanter, S. Ahmed, D. Pau, et al., MLPerf tiny benchmark, 2021, arXiv preprint arXiv:2106.07597.
[133] M. Mazumder, C. Banbury, X. Yao, B. Karlaš, W.G. Rojas, S. Diamos, G. Diamos, L. He, D. Kiela, D. Jurado, D. Kanter, R. Mosquera, J. Ciro, L. Aroyo, B. Acun, S. Eyuboglu, A. Ghorbani, E. Goodman, T. Kane, C.R. Kirkpatrick, T.-S. Kuo, J. Mueller, T. Thrush, J. Vanschoren, M. Warren, A. Williams, S. Yeung, N. Ardalani, P. Paritosh, C. Zhang, J. Zou, C.-J. Wu, C. Coleman, A. Ng, P. Mattson, V.J. Reddi, DataPerf: Benchmarks for Data-Centric AI Development, 2022, http://dx.doi.org/10.48550/ARXIV.2207.10062, URL https://arxiv.org/abs/2207.10062.
[134] Z. Jiang, W. Gao, L. Wang, X. Xiong, Y. Zhang, X. Wen, C. Luo, H. Ye, X. Lu, Y. Zhang, S. Feng, K. Li, W. Xu, J. Zhan, HPC AI500: A Benchmark Suite for HPC AI Systems, in: C. Zheng, J. Zhan (Eds.), Benchmarking, Measuring, and Optimizing, Springer International Publishing, Cham, 2019, pp. 10–22.
[135] The Standard Performance Evaluation Corporation (SPEC), 2022, https://www.spec.org/.
[136] C. Bienia, Benchmarking Modern Multiprocessors (Ph.D. thesis), Princeton University, 2011.