

Noise in the Clouds: Influence of Network Performance Variability on Application Scalability

Daniele De Sensi, Tiziano De Matteis, Konstantin Taranov,
Salvatore Di Girolamo, Tobias Rahn, and Torsten Hoefler

Department of Computer Science, ETH Zurich, Switzerland
 {first-name.last-name}@inf.ethz.ch

Abstract—Cloud computing represents an appealing opportunity for cost-effective deployment of HPC workloads on the best-fitting hardware. However, although cloud and on-premise HPC systems offer similar computational resources, their network architecture and performance may differ significantly. For example, these systems use fundamentally different network transport and routing protocols, which may introduce *network noise* that can eventually limit the application scaling. This work analyzes network performance, scalability, and cost of running HPC workloads on cloud systems. First, we consider latency, bandwidth, and collective communication patterns in detailed small-scale measurements, and then we simulate network performance at a larger scale. We validate our approach on four popular cloud providers and three on-premise HPC systems, showing that network (and also OS) noise can significantly impact performance and cost both at small and large scale.

Index Terms—cloud; HPC; network noise; scalability;

I. INTRODUCTION

Due to flexibility and cost-effectiveness, running HPC applications in the cloud has become an appealing solution and a potential alternative to on-premise systems [1], [2]. Scientific applications from different domains already run on the cloud, including multiphysics simulations [3], [4] and biomedical applications [5], [6].

One of the main advantages of cloud computing is the possibility to run an application on the most appropriate computational resources in a cost-effective way. Instances that can be deployed in the cloud come with a wide variety of architectural characteristics in terms of memory, CPUs, accelerators, and network bandwidth. On the CPU side, it is possible to select between different processors, with different numbers of cores, clock frequency, and architecture, ranging from commercial off-the-shelf Intel and AMD processors to custom ARM processors like the ARM Graviton processor deployed by AWS [7]. Cloud providers also offer a wide choice of accelerators that includes different types and generations of GPUs [8], TPUs (*Tensor Processing Units*) [9], and FPGAs [10]. Similarly, different instances provide different network bandwidths. Users can deploy instances with 100 Gb/s networks on most major cloud providers and, in some cases, even 200 Gb/s and 400 Gb/s instances (on Azure and AWS, respectively). Finally, cloud vendors frequently deploy new hardware, differently from on-premise HPC systems, where compute resources have life-cycles spanning multiple years.

However, all this flexibility comes at a cost. Although we can expect minor differences in the compute performance between an HPC instance in the cloud and an equivalent server in an on-premise HPC system [11], [12], the network performance can significantly differ. Indeed, in some cases, the network connecting those instances in the cloud significantly differs from a traditional HPC network. For example, packets might be routed using ECMP [13], [14], [15] in a congestion-oblivious way, and can thus experience a higher latency if multiple network flows are mapped on the same paths [16], [17], [18]. On the contrary, HPC systems often deploy adaptive routing to react more promptly to congestion in the network [19], [20]. Also, differently from most HPC systems, some providers do not use *Remote Direct Memory Access* (RDMA), or run instances on tapered networks [15]. All these factors can contribute to increase network latency, decrease network bandwidth, and increase *network noise* [21], [22], [23], [24], [25] (i.e., performance variability induced by the use of the network). This limits the scalability and tampers cost-effectiveness. Although HPC applications can scale up to 42 million cores [26] on on-premise HPC systems, it is still not clear how far HPC applications could scale on the cloud. Traditionally, cloud environments have been considered a good match for loosely coupled or embarrassingly parallel workloads, but network performance has been seen as one of the main bottlenecks preventing their adoption for tightly coupled computations [27], [11], [28], [29], [30], [31].

Assessing the network performance and the impact of noise on scalability is even more relevant if we consider that the gap between compute and network performance increases. For example, *from 2010 to 2018, the computational throughput of the Top 500 HPC systems* [32] *increased by 65x, while the off-node communication bandwidth only increased by 4.8x* [33], [34]. Thus we expect, in the future, network performance to be even more relevant for HPC applications running on the cloud.

In this work, we focus on network performance and noise, assessing the impact on performance, scalability, and cost of tightly-coupled HPC communication patterns at scale. Because collecting statistically sound measurements at the scale of thousands of HPC VMs would be too expensive (and on some cloud providers not even feasible), we first perform detailed network performance and noise measurement at small scale.

TABLE I
ANALYZED SYSTEMS: FOR EACH OF THEM WE DETAIL THE CPU, MEMORY AND NETWORK CHARACTERISTICS. *C* INDICATES THE NUMBER OF PHYSICAL CORES. INSTANCE COSTS AS REFERRED TO THE JULY, 18 2022 FOR THE EAST US AVAILABILITY ZONE.

SYSTEM	INSTANCE TYPE	CPU	MEMORY	PER HOUR INSTANCE COST (COMMITTED)	PER HOUR INSTANCE COST (ON-DEMAND)	BANDWIDTH	NETWORK	ROUTING	TRANSPORT PROTOCOL	
AWS	Normal	c5.18xlarge	2x18C Intel Xeon Platinum 8124M @ 3GHz	144 GB	1.34 USD	3.06 USD	25 Gb/s	Fat Tree [14]	ECMP [14]	SRD [14]
	HPC (Metal)	c5n.metal	2x18C Intel Xeon Platinum 8124M @ 3GHz	192 GB	1.475 USD	3.88 USD	100 Gb/s	Fat Tree [14]	ECMP [14]	SRD [14]
	HPC	c5n.18xlarge	2x18C Intel Xeon Platinum 8124M @ 3GHz	192 GB	1.475 USD	3.88 USD	100 Gb/s	Fat Tree [14]	ECMP [14]	SRD [14]
AZURE	Normal	F72s_v2	36C Intel Xeon Platinum 8370C/8272CL/8168	144 GB	1.116 USD	3.045 USD	30 Gb/s	Fat Tree [35]	ECMP [35]	N.A.
	HPC	HC44rs	2x22C Intel Xeon Platinum 8168 @ 2.70GHz	352 GB	2.218 USD	3.168 USD	100 Gb/s	Non-Blocking Fat Tree [36]	Static/Adaptive [37]	InfiniBand [36]
	HPC (200 Gb/s)	HB120rs_v2	2x60C AMD Epyc 7V12 @ 2.45 GHz	456 GB	1.8 USD	3.6 USD	200 Gb/s	Non-Blocking Fat Tree [36]	Static/Adaptive [37]	InfiniBand [36]
GCP	Normal	c2-standard-60	2x15C Intel Cascade Lake @ 3.10GHz	240 GB	1.25 USD	3.1321 USD	32 Gb/s	Jupiter (3:1 Blocking Fat Tree) [15]	ECMP [15]	TCP/IP + Intel QuickData [38]
	HPC	c2-standard-60	2x15C Intel Cascade Lake @ 3.10GHz	240 GB	2.148 USD	4.03 USD	100 Gb/s	Jupiter (3:1 Blocking Fat Tree) [15]	ECMP [15]	TCP/IP + Intel QuickData [38]
ORACLE	Normal	VM.Optimized3.Flex	18C Intel Xeon Gold 6354 @ 3GHz	256 GB	N.A.	1.188 USD	40 Gb/s	Non-Blocking Fat Tree [39]	N.A.	N.A.
	HPC (Metal)	BM.Optimized3.36	2x18C Intel Xeon Gold 6354 @ 3GHz	512 GB	N.A.	2.712 USD	100 Gb/s	Non-Blocking Fat Tree [39]	N.A.	RoCEv2 [40]
DAINT	HPC (Metal)	-	2x18C Intel Xeon E5-2695 v4 @ 2.10GHz	64 GB	1.02 USD [41]	1.73 USD [41]	82 Gb/s	Cray Aries (Dragonfly) [42]	Per-Packet Adaptive [42]	FMA [42]
ALPS	HPC (Metal)	-	2x64C AMD EPYC 7742 @ 2.25GHz	256 GB	N.A.	N.A.	100 Gb/s	HPE Cray Slingshot (Dragonfly) [19]	Per-Packet Adaptive [19]	RoCEv2 [19]
DEEP-EST	HPC (Metal)	-	2x12C Intel Xeon Gold 6146 @ 3.20GHz	192 GB	N.A.	N.A.	100 Gb/s	Mellanox InfiniBand EDR (Fat Tree) [43]	Static/Adaptive	Infiniband [43]

On one side, we analyze this data to spotlight differences in network performance and noise between different cloud and on-premise HPC systems. On the other side, we use this data to calibrate the LogGOPSim simulator [44], [45], and to simulate the scalability and cost at a larger scale (up to 16K HPC VMs).

We define the concepts of *latency noise* and *bandwidth noise*, and assess the network performance and its impact on scalability of HPC and normal instances of four major cloud providers and of three on-premise systems (with different network technology). We also assess OS noise (i.e., performance variability introduced by OS processes), and we show how different type of noise impact application performance and cost both at small and large scale, for both latency- and bandwidth-dominated communication patterns.

We describe in Sec. II the main characteristics of HPC cloud solutions, in Sec. III we analyze the network performance of both cloud and on-premise HPC systems, with a focus on OS and network noise in Sec. IV. Then, we simulate how noise affects performance at scale in Sec. V, and discuss related work in Sec. VII. Eventually, Sec. VIII draws conclusions.

II. HPC IN THE CLOUD

In this section we measure and analyze the network performance of HPC systems in the cloud at a small scale, to understand better some peculiarities and limitations of those systems. In this paper, we analyze four of the major cloud providers: Amazon AWS [46], Google GCP [47], Microsoft Azure [48], and Oracle Cloud [49]. We also analyze three on-premise HPC systems: Piz Daint [50] (referred as *Daint* in the following) and Alps [51], both deployed at the Swiss National Supercomputing Centre, and DEEP-EST [43], deployed at the Jülich Supercomputing Centre. We analyze cloud instances of different types, including HPC instances (with different network bandwidth) and normal compute instances. We outline the different analyzed systems, instance types, and their main characteristics in Table I. In the following we analyze in detail the different instance types (Sec. II-A), their network features (Sec. II-B), and their cost (Sec. II-C).

A. Instances, CPUs, and OS

In the following, with the term *HPC instances*, we refer to those instances providing at least 100 Gb/s networking.

For AWS, we evaluate both bare-metal and non bare-metal HPC instances. Azure and GCP provide only non bare-metal HPC instances, whereas Oracle only provides bare-metal HPC instances. To have a fair comparison, we selected instance types with similar CPUs when possible. We used Intel CPUs on all the cloud instances except for the 200 Gb/s instances of Azure, which only have AMD EPYC CPUs. For normal instances, we selected those that provide a similar network bandwidth and core count. For completeness, we also report the amount of RAM memory on each instance type.

All four providers guarantee that HPC instances are run on separate physical servers. For the normal instances we selected CPUs with a high core count to have them allocated on two separate servers. This is necessary to ensure that when measuring network performance the two VMs are actually using the network. For the cloud providers we report in the *Instance Type* column the name of the instances we used.

On all cloud providers, we use the virtual machine (VM) images and operating system suggested for the HPC instances. These were: Amazon Linux 2 on AWS [52], CentOS 7.7 on Azure [53], CentOS 7.9 on GCP [54], and Oracle Linux 7.9 on Oracle. Daint and Alps run a Cray Linux Environment (CLE) OS based on SUSE Linux Enterprise Server v15.2, and DEEP-EST runs Rocky Linux v8.5.

B. Network

The four cloud providers and the DEEP-EST system deploy a fat tree topology. According to the most recent documentation we found, Azure, Oracle, and DEEP-EST deploy a non-blocking network [36], [55], GCP a 3:1 blocking network [15], whereas we did not find any additional detail on network over- or under-provisioning for AWS. Both AWS and GCP use ECMP routing [13], Azure employs adaptive routing [37] for HPC instances, and for Oracle we did not find any information on routing. The routing protocol plays a crucial role in network performance. For example, ECMP is congestion oblivious and might suffer from flow collisions [16], [17], [18], increasing the network bandwidth variability (see Sec. IV-C). Daint and Alps deploy a dragonfly interconnect (Cray Aries [42] and Slingshot [19] respectively) with adaptive routing.

Each of the evaluated cloud providers uses a different transport protocol. AWS provides its proprietary RDMA-like protocol called SRD (*Scalable Reliable Datagram*) [14], which resembles in some aspects InfiniBand verbs [56]. It provides reliable out-of-order delivery of packets and uses a custom congestion control protocol. The AWS Nitro Card [57] implements the reliability layer, and the *Elastic Fabric Adapter* (EFA) provides OS-bypass capabilities. To react to congestion, SRD monitors the round trip time (RTT) and forces packets to be routed differently by changing some of the fields used by ECMP to select the path. This approach is probabilistic and might allow avoiding congested paths, but, differently than truly adaptive routing, it does not allow selecting the least congested path nor any specific path.

Azure and DEEP-EST use RDMA through InfiniBand [36], Oracle uses *RDMA over Converged Ethernet* (RoCEv2) [40],

whereas GCP does not use RDMA and relies on TCP/IP. To minimize data movement overheads, GCP uses Intel's QuickData DMA Engines [58] to offload payload copies of larger packets. Daint uses a proprietary RDMA protocol [42] (FMA), whereas Alps uses RoCEv2 [19].

C. Cost

Table I shows the per-hour cost charged to the user as of July 18, 2022. For the cloud systems, we report the cost for the East US availability zone. We consider both the cost for a committed 3-years usage with upfront payment and the on-demand cost without any minimum commitment. Please note that 3-years is the maximum commitment allowed on those providers (and that leads to the lowest per-hour cost), whereas when having no commitments we have the highest per-hour cost. For Daint, we report the cost for a minimum usage of 10,000 compute hours, as well as the on demand cost, both for non-academic partners. Academic partners have discounted rates and this would otherwise lead to an unfair comparison. For Alps and DEEP-EST there is no publicly available information on the per-hour cost.

On AWS, the main difference between the normal and HPC instances we selected is the support for *Elastic Fabric Adapter* (EFA), which provides the 100 Gb/s networking. Thus, we can estimate the 3-years committed cost of the high-performance networks at around 0.135 USD per hour per VM, and an on-demand cost of 0.82 USD. Similarly, we selected the same instance type for normal and HPC instances on GCP. The only difference is that we enabled the so-called *Tier 1* network on the HPC instance, which provides 100 Gb/s network bandwidth. On GCP, we can thus estimate the cost of the HPC network at around 0.9 USD per hour per VM [59] (both for the committed and on-demand cost). Unfortunately, Azure and Oracle do not provide the same instance in HPC and non-HPC flavors, and it is thus not possible to isolate the cost of the HPC network from the rest. Also, we observe that whereas the on-demand cost of 100 Gb/s instances is lower than the cost of 200Gb/s instances, this is not true for the 3-years committed usage. Indeed, at the time of the writing, committing for a 3-years usage led to a 30% discount for 100 Gb/s instances, and to a 50% discount for the 200Gb/s ones.

III. NETWORK PERFORMANCE

We measure network performance using the *Netgauge* tool [60], that provides detailed, sample-by-sample measurements (fundamental for estimating network noise in Sec. IV). We used the *Message Passing Interface* (MPI) backend and, on each system, the MPI library recommended by the provider¹.

a) Methods: We created an account on each provider, and used our own funding and/or academic credits, without coordinating with the providers. The clusters have been created and tuned following the guidelines publicly available

¹On Azure we used HPC-X v2.8.3 on HPC instances and Open MPI v4.1.2 on normal instances. We used Open MPI v4.1.1 on AWS, Intel MPI v2018.4.274 on GCP, Open MPI v4.0.4 on Oracle, Cray MPICH v7.7.18 on Daint, Cray MPICH v8.1.12 on Alps, and Open MPI v4.1.3 on DEEP-EST.

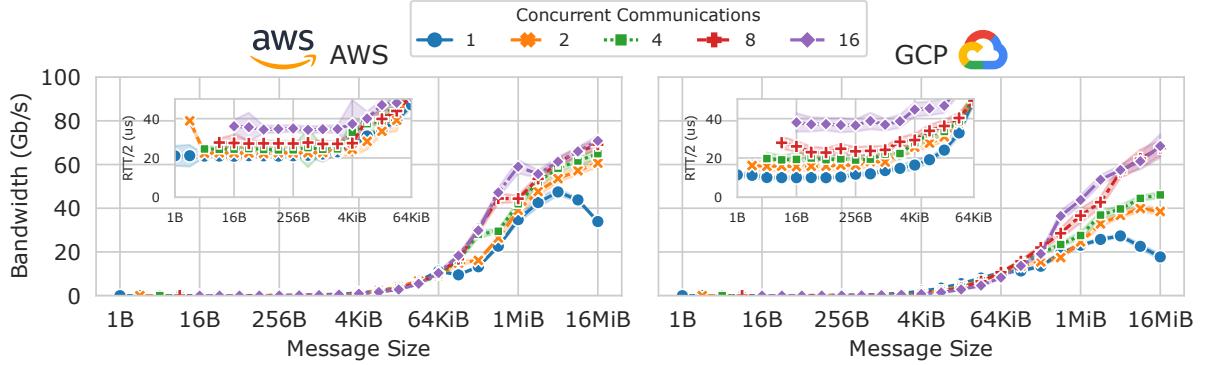


Fig. 1. Bandwidth for HPC instances as a function of message size and number of concurrent connections between the two servers. Inner plots show RTT/2 for small messages.

in the documentation of the cloud providers. After running the benchmarks, we contacted the leads of the cloud business of each of the providers, sharing a draft of the paper with them. They assessed the correctness of our evaluation, and we integrated their feedback in the paper. Only in one case we improved the performance by applying a technique not described in the publicly available documentation, that we describe in the text (see the comment about the `FI_EFA_TX_MIN_CREDITS` in Sec. III-A). On all the providers, if not specified otherwise, we allocated the two VMs (or the two servers) on the same rack. The only exception is Oracle, where it is not possible to explicitly control the allocation. We analyze in detail the impact of allocation on performance and noise in Sec. IV.

A. Bandwidth Saturation

All the four analyzed cloud providers claim a 100 Gb/s bandwidth on Intel-based HPC instances. However, this is true only under certain conditions. For example, AWS documents a maximum per-message bandwidth of 25Gb/s [61]. Even if not explicitly documented, we observed similar limitations on GCP. One possible reason justifying this behavior is that even if the instance exposes a single 100 Gb/s NIC, it might be equipped with multiple 25Gb/s NICs (or a multi-port NIC). While some providers explicitly documented this for non-HPC instances, the specific configuration is often unclear for HPC ones.

For this reason, we can expect a higher bandwidth when sending a message over multiple connections. To assess if this is the case, we run a ping-pong benchmark between two nodes. We establish multiple concurrent connections between the two nodes, by running multiple processes per node and letting each pair of processes send/receive disjoint parts of the message. For example, a 16MiB pingpong with 16 processes per node runs 16 concurrent ping-pongs between 16 processes on the first node and 16 processes on the second node, each with a 1MiB message.

We report the results of this experiment for AWS and GCP in Figure 1. We report the bandwidth as the message size divided by half the round trip time (RTT/2), and the inner plots

show the RTT/2 for small messages. Each point in the plot is the average over 1000 runs, whereas the band around the point represents the standard deviation. We do not report the results for the other systems since they can saturate the bandwidth even with a single connection (we show results in the next section). On AWS we increased the bandwidth by increasing the maximum number of in-flight packets to 1024 (by setting the `FI_EFA_TX_MIN_CREDITS` environment variable).

On both AWS and GCP the bandwidth increases when increasing the number of concurrent communications (up to 80Gb/s with 16 processes per node). Also, when using a single connection, the bandwidth drops for messages larger than 4MiB. This is caused by a more-than-linear increase in last level cache (LLC) misses, that we measured by using the `perf` tool. For example, on AWS, we observe a 4× increase in LLC misses when going from 1MiB to 4MiB messages, but a 8× increase when moving from 4MiB to 16MiB messages. This effect is not present when using more concurrent communications because the message is split among the processes, each transmitting a smaller message.

We also observe that having more processes per node increases the RTT of small messages, due to additional overhead and contention on the NIC access. For this reason, only large messages should be sent with multiple concurrent connections. Instead of having more processes sending a part of the message each, we could have a single process sending multiple smaller messages. For example, some MPI libraries provide the possibility to *stripe* messages transparently over multiple connections (e.g., by using the `btl_tcp_links` command line flag on Open MPI [62]). However, we did not observe any performance improvement compared to the single connection case.

Observation 1: *On AWS and GCP, the peak bandwidth on a single connection is 50Gb/s and 30Gb/s respectively. A bandwidth of 80Gb/s can only be reached by forcing messages to be concurrently sent/received by/from multiple processes on different connections.*

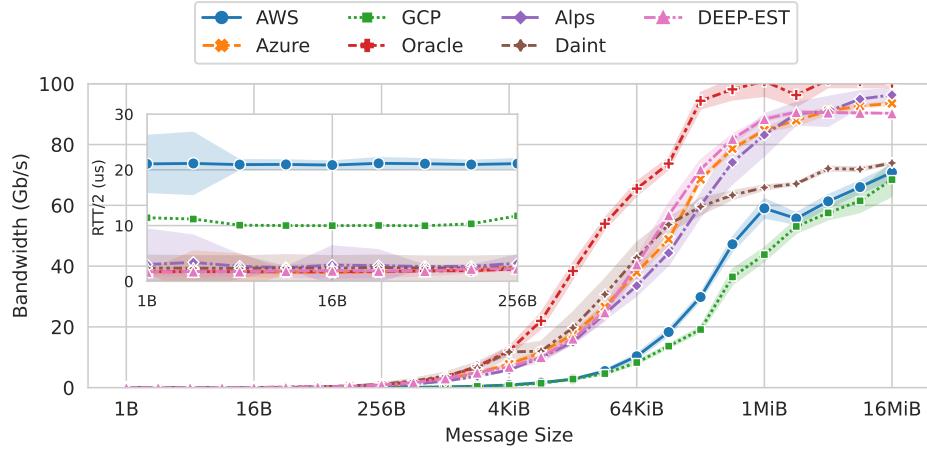


Fig. 2. Unidirectional bandwidth on different providers as a function of the message size. Inner plots show RTT/2 for small messages. For AWS and GCP we report the results with optimal number of connections (1 for minimizing RTT/2 for small messages, and 16 for maximizing bandwidth on large messages).

B. Unidirectional Bandwidth and Latency

Figure 2 shows the RTT/2 and bandwidth of different providers as a function of the message size. For the cloud providers we selected the 100 Gb/s instances (bare-metal when available). For AWS and GCP, we report the RTT results with a single connection (lowest RTT on small messages), and for bandwidth, the results with 16 concurrent connections (highest bandwidth on large messages).

Regarding the latency (i.e., the RTT/2 for 1 byte messages), we observe that Azure, Oracle, and the on-premise systems exhibit a 1-2 microseconds latency for HPC instances. On the other hand, both AWS and GCP are characterized by much higher latencies (20 and 10 microseconds respectively). Concerning the bandwidth, Azure, Oracle, Alps, and DEEP-EST achieve a bandwidth higher than 90 Gb/s for 16MiB messages. On Daint, we measured 75 Gb/s peak bandwidth for 16MiB messages (NICs on Daint have an injection bandwidth of 82 Gb/s [42]). AWS and GCP reach a peak bandwidth of circa 70 Gb/s (using 16 concurrent connections). Oracle achieves the 90% of the declared bandwidth with 256KiB messages, Alps, Azure and DEEP-EST with 2MiB messages, Daint with 4MiB messages, and AWS and GCP only achieve the 70% of the declared bandwidth with 16MiB messages.

Observation 2: Azure and Oracle achieve network latency and bandwidth comparable to that of on-premise HPC systems. On the other hand, GCP and AWS achieve 25% lower bandwidth and 10x higher latency.

C. Bidirectional Bandwidth

To measure the bidirectional bandwidth, we perform two simultaneous ping-pongs between two nodes, with each ping-pong starting from a different node. In Figure 3, we report the results of this experiment, and we compare the peak unidirectional and bidirectional bandwidth with 16MiB messages. For

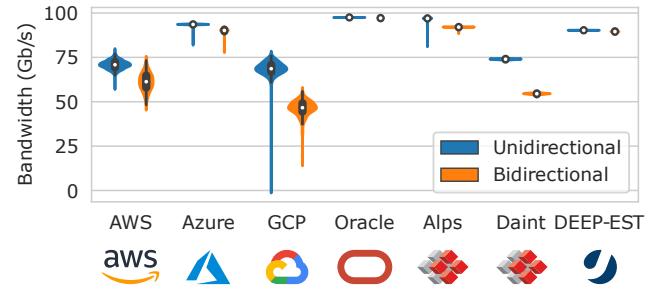


Fig. 3. Peak unidirectional and bidirectional bandwidth.

both AWS and GCP, we use 16 concurrent connections. In some cases, we observe a peak bidirectional bandwidth lower than the peak unidirectional bandwidth. For example, on Daint this is caused by message requests and responses sharing the same data path, decreasing the peak theoretical bandwidth per direction to 64 Gb/s [42].

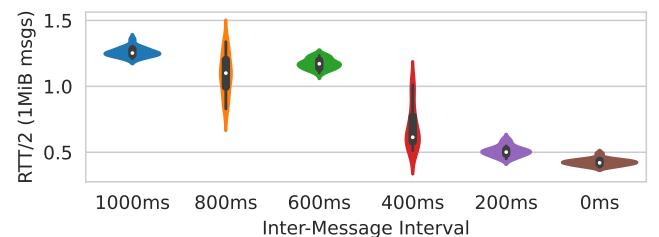


Fig. 4. Distribution of RTT/2 (ms) of 1MiB transfers on GCP HPC instances, for different inter-message intervals.

D. Traffic Burstiness

We now investigate the impact of traffic burstiness on network performance. To assess this, we execute a 1MiB ping-pong between two nodes, varying the *inter-message interval*,

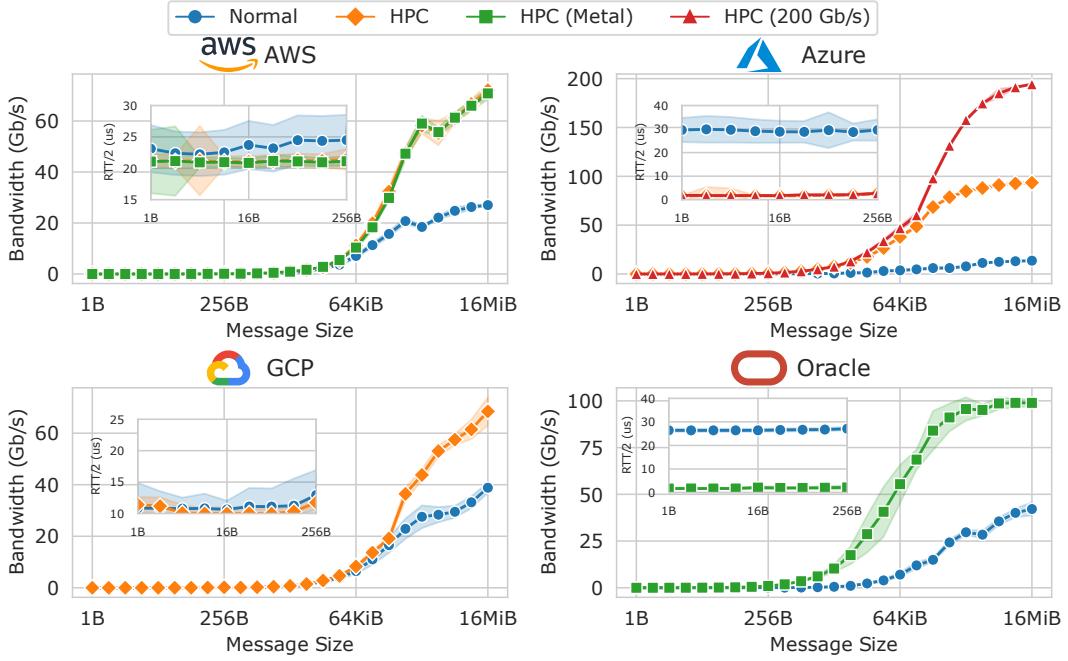


Fig. 5. Unidirectional bandwidth for the different instance types described in Table I, organized by provider. Inner plots show the RTT/2 for small messages. Note that each plot uses a different scale.

i.e., the time between two subsequent message transmissions between 0 and 1 second. To exclude any pipelining effect, the benchmark waits for a message to be completely received before sending the next one. We repeat each experiment for 20 iterations, with 10 warm-up iterations. We did not observe any impact of burstiness on the performance, except for GCP, for which we report the results in Figure 4. We show on the X axis the interval between two subsequent messages, and on the Y axis the RTT/2 (milliseconds).

We observe how, when the time interval between subsequent messages is one second, a 1MiB message requires around 1.5 milliseconds to be transferred from the source to the destination. On the other hand, when we decrease the inter-message interval, the RTT starts decreasing, and, eventually, the RTT/2 becomes lower than 0.5 milliseconds. We observed this behavior consistently in multiple runs, in different days, and at different times of the day.

Our initial assumption was that this could be related to the Andromeda network virtualization stack, used by GCP for forwarding packets over the network [38]. To scale on very large networks, and avoid storing thousands of VM-to-VM forwarding rules on each VM, the Andromeda VM host stack sends all the packets for which it does not have a route to *Hoverboard* gateways. If the Andromeda control plane detects that a flow exceeded some bandwidth usage threshold, it installs direct VM-to-VM forwarding rules in the VM host stack, so that high-bandwidth flows are forwarded directly to the destination VM without the need to traverse the *Hoverboard* gateways to resolve the forwarding rule. Although

this works well for bandwidth-intensive flows, bursty flows might not trigger the installation of a forwarding rule in the software stack of the source VM, thus incurring in the extra latency required to traverse a *Hoverboard* gateway.

However, after a discussion with GCP engineers (that were able to reproduce and confirm the issue) we believe that this is not caused by *Hoverboard*. Indeed, the same behaviour also happens if the two communicating process run on the same node, thus using shared memory rather than the network for communicating. We also exclude issues with the MPI implementation, since we observed the same behavior when communicating directly using TCP rather than MPI. This is also not caused by CPU power saving features, that were disabled during the tests. We are currently investigating, with the help of GCP engineers, the reasons for this behaviour, that are likely related to virtualization.

Observation 3: *On GCP, large delays between messages can increase the RTT/2 up to 3x compared to the case where messages are sent back to back.*

E. HPC vs. Normal Compute Instances

Figure 5 reports the unidirectional bandwidth and RTT/2 for the different instance types and cloud providers described in Table I. For GCP and AWS we report the bandwidth when using 16 concurrent connections. For Azure normal compute instances, we achieved the highest bandwidth (14 Gb/s) with two concurrent connections, and we observed bandwidth degra-

tion when running more than two concurrent connections. We believe that on Azure normal instances concurrent connections are needed because each VM uses multiple NICs with lower bandwidth, as documented by Azure [63]. The Azure 200 Gb/s instances reach the peak bandwidth with 16MiB messages. Also, we observe that normal compute instances on Azure are characterized by the lowest bandwidth, whereas AWS normal instances can achieve a 25Gb/s bandwidth, Oracle normal instances achieve a bandwidth of 40Gb/s, and GCP achieve even higher bandwidth than that declared (40Gb/s versus 32Gb/s).

Regarding the latency, on AWS, HPC instances are characterized by a marginally lower RTT/2 compared to normal instances. On Azure, we observed a 30 microseconds RTT/2 on normal instances, much higher than that observed on the HPC instances (around 1-2 microseconds). The same holds for Oracle, where on normal instances we observed a latency of almost 30 microseconds. On GCP, we observed no difference in the latency between normal and HPC instances.

Observation 4: *On AWS and GCP, HPC instances communicate with the same latency as normal instances, whereas on Azure and Oracle HPC instances communicate with a latency 10-20x lower than normal instances.*

IV. NETWORK AND OS NOISE

Application performance can significantly vary across different runs due to effects such as OS and network noise [34], [64], [65]. To analyze how these different types of noise can affect the performance of a large scale system, we now describe how they affect the different terms that contribute to the time needed to deliver a message according to the LogGP model [66]. We focus on this model because, although simple, it captures the fundamental aspects of network communications, and allows us to implement a solid simulation methodology (discussed in Sec. V). LogGP models the time needed for sending a message of s bytes as $T(s) = 2o + L + (s-1)G$, where o is the overhead for sending (and receiving) a message, L is the network latency, and G is the gap between the transmission of two subsequent bytes (i.e., the inverse of the network bandwidth). The model also has an additional parameter g representing the minimum time between the transmission of two subsequent messages. This parameter is not shown in the formula above, because it represents the transmission time of a single message only.

To outline the model visually, we show in Figure 6 these parameters for a scenario where a node transmits three messages. The first two messages are composed of four bytes, while the third consists of a single byte. Whereas the transmission of the first message is not affected by any type of noise, the transmission of the second message experiences both *bandwidth noise* and *latency noise*. Bandwidth noise occurs in the network, and affects the gap per byte G , slowing down the transmission of subsequent bytes (or packets) (①). Also latency noise (②) occurs in the network, and manifests itself

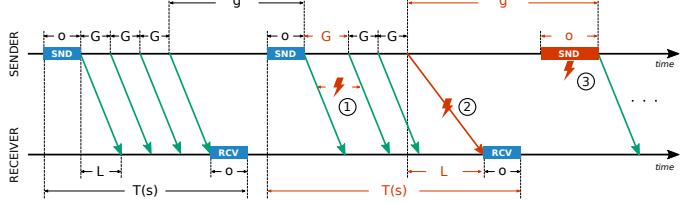


Fig. 6. Impact of the different types of noise on the LogGP parameters. ① = bandwidth noise, ② = latency noise, ③ = OS noise.

as an increase in the latency L . It delays the transmission of some bytes (or packets) of the message, and might be caused by transient congestion in the network. The main difference between latency and bandwidth noise is in the duration of the noise events. For example, ECMP routing can cause persistent bandwidth noise due to mapping of multiple network flows on the same paths. Last, OS noise occurs on the host (③), can affect both the o and g parameters and, in general, can also cause an increase in the transmission time of a message. The impact of OS noise on the scalability of HPC systems has been largely studied [64], [65], [67], but it is still not clear what is its impact on cloud HPC systems, or whether network noise is the major component of cloud systems noise.

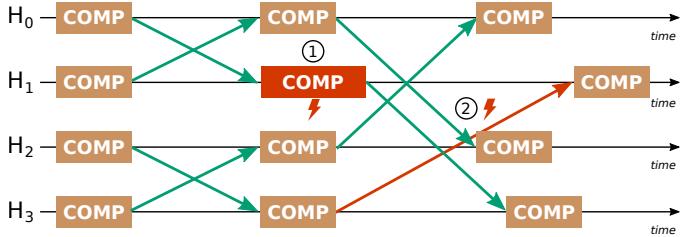


Fig. 7. Noise affecting a tightly coupled butterfly-structured collective operation. ① = OS noise, ② = latency noise

Moreover, the larger the application scale, the higher the probability that some data transmissions experience either OS or network noise. This can be an issue in tightly coupled applications, where it is enough to delay one single process to slow down the entire application, thus limiting its scalability. Moreover, multiple noise events can either overlap, or sum up and amplify. We exemplify this in Figure 7, showing a few steps of a collective operation performing a butterfly communication pattern. In this example, one process is affected by OS noise, thus delaying the transmission of its data (①), while another process is affected by latency noise (②). Because these two noise events overlap, the application is only delayed by the maximum between the two. In general, non-overlapping noise events can accumulate and significantly increase the application execution time. In the remaining part of this section, we analyze the noise events we observed on the systems of Table I. Then, in Sec. V we simulate how these events affect applications at scale.

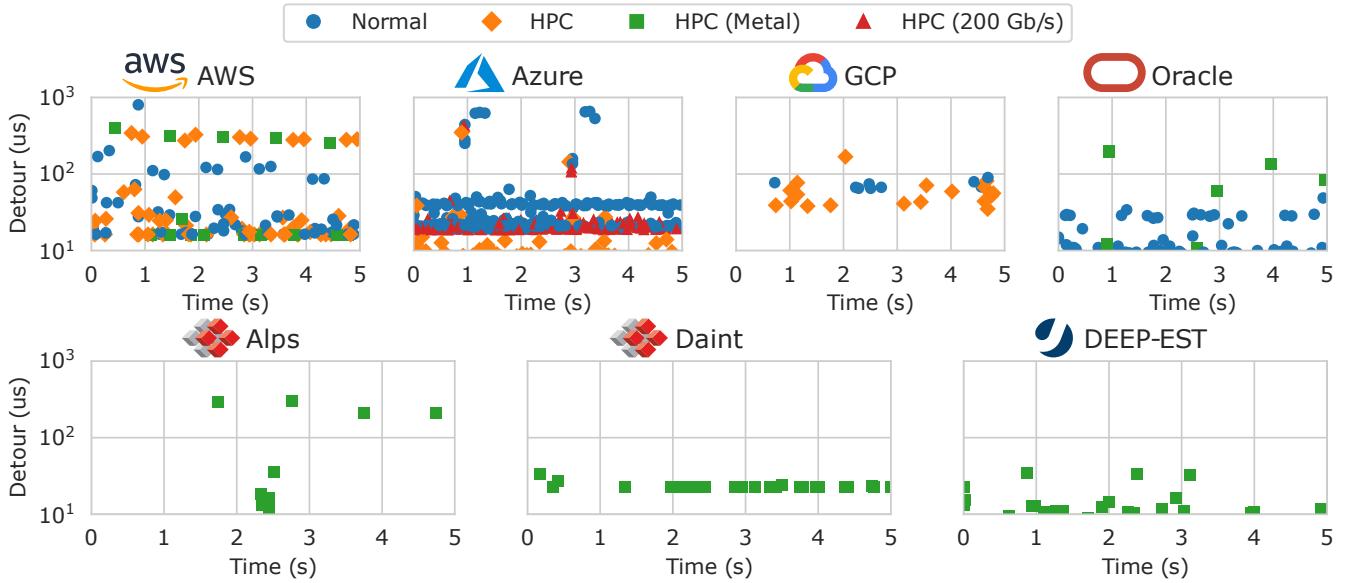


Fig. 8. OS noise for the different instance types described in Table I for the different providers.

A. OS Noise

We measured OS noise by running the *selfish detour* benchmark provided by Netgauge, using the same parameters suggested by the authors [65]. The benchmark measures perturbation introduced by the OS, and has been successfully used in the past to assess OS noise of some on-premise HPC systems [65]. It runs a tight loop and records all the iterations larger than $9 \cdot t_{min}$, where t_{min} is the minimum measured time required to complete one iteration. The benchmark runs until a predefined number of iterations are recorded. Figure 8 shows detour durations over time for a 5 seconds window, for all considered providers. To improve the readability of the plots, we only report the largest samples (top 1%). It is worth remarking that those largest samples are exactly those representing the noise and limiting applications scalability [65].

We observe that on-premise systems have the lowest OS noise, as well as HPC bare-metal instances on AWS and Oracle. On the other hand, on AWS and Azure, HPC and normal instances experience noise with high intensity and high frequency, whereas GCP experience noise with medium frequency and low intensity. On Azure, normal and 200 Gb/s instances experience noise with higher frequency than the 100 Gb/s HPC instances.

Observation 5: When available, bare-metal HPC instances are characterized by a lower OS noise than non bare-metal instances, and are comparable to on-premise HPC systems. Normal instances are characterized by OS noise with high frequency.

B. Latency Noise

We now evaluate the impact of latency noise by running a 1-byte ping-pong for one hour. Generally, the performance of a multi-tenant computing system might change depending on the time of the day (due to the different system utilization). We collected data for 24 consecutive hours on all the analyzed systems (not shown here due to space constraints), and we did not observe significant intra-day variability. For consistency, we report here the data collected at 5 PM in the local time of the cluster. Performance might also change depending on the distance between the two nodes in the network. For this reason, we analyze the noise for different node distances. On AWS and GCP it is possible to specify whether the VMs in a cluster must share the same rack (and thus the same network switch) or not. We did the same on the three on-premise systems for consistency and fairness. On Azure, unfortunately, to use the high-bandwidth network, HPC VMs must always share the same rack. On Oracle it is not possible to specify or check the allocation. However, to simplify the exposition of the results, we assume that the two VMs share the same rack.

Figure 9 reports the results of our latency noise assessment on the different providers for different node distances for 100 Gb/s HPC instances (bare-metal when applicable). We show the latency normalized with respect to the minimum latency (for completeness, we report in Table II the minimum and average latency). To improve the readability of the plot, we only report the largest 0.1% measurements.

We observe that, when the two nodes are on the same rack, Alps experiences latency noise with the highest intensity, with samples experiencing a latency more than 128 times higher than the minimum. Also, on Oracle we observed one measurement equal to 59 milliseconds, 35000 times larger than the minimum latency. The other systems instead experience

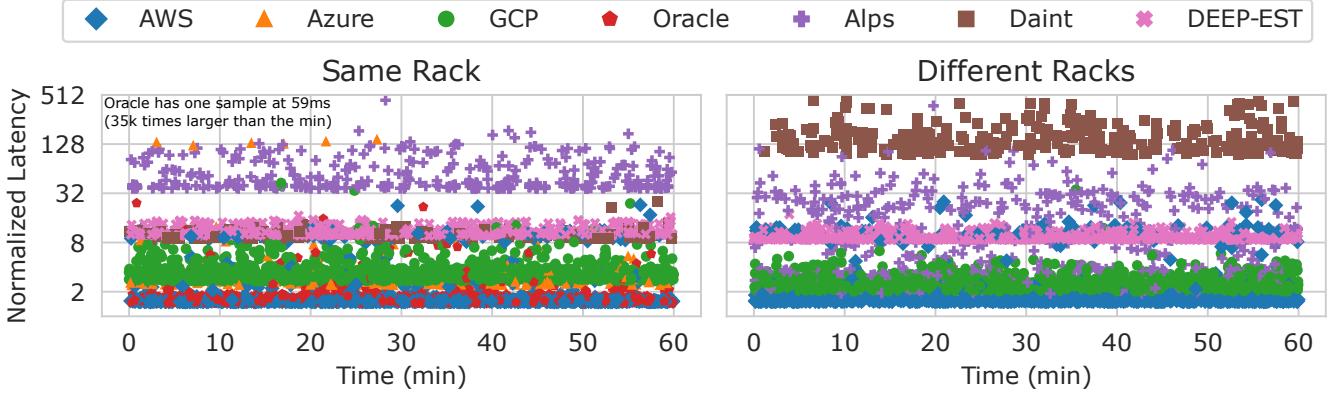


Fig. 9. Latency noise for different node distances for 100 Gb/s HPC instances. Base latency is reported in Table II.

TABLE II
MINIMUM AND AVERAGE LATENCY AND BANDWIDTH FOR THE DIFFERENT PROVIDERS, INSTANCE TYPES, AND NODE DISTANCES.

	AWS		Azure			GCP		Oracle		Alps	Daint	DEEP-EST
	Normal	HPC (Metal)	Normal	HPC	HPC (200 Gb/s)	Normal	HPC	Normal	HPC (Metal)	HPC (Metal)	HPC (Metal)	HPC (Metal)
SAME RACK	Min. Lat. (us)	19.28	16.79	26.11	1.50	1.70	9.42	8.46	24.68	1.66	2.13	1.19
	Mean Lat. (us)	23.11	18.97	29.59	1.65	1.84	10.64	9.98	26.43	1.72	3.01	2.39
	Max. Band. (Gb/s)	27.32	78.74	7.42	93.92	194.48	45.45	75.75	11.20	97.53	97.14	74.37
	Mean Band. (Gb/s)	27.02	70.84	7.28	93.51	194.25	38.82	68.45	8.39	97.50	96.32	73.93
DIFFERENT RACKS	Min. Lat. (us)	22.46	17.20	N.A.	N.A.	N.A.	12.39	14.90	N.A.	N.A.	2.66	1.19
	Mean Lat. (us)	27.57	19.26	N.A.	N.A.	N.A.	15.02	16.66	N.A.	N.A.	2.90	3.33
	Max. Band. (Gb/s)	30.52	77.72	N.A.	N.A.	N.A.	34.84	70.11	N.A.	N.A.	96.15	75.24
	Mean Band. (Gb/s)	30.14	67.02	N.A.	N.A.	N.A.	30.67	65.71	N.A.	N.A.	96.00	74.59

noise with lower intensity. When the two nodes are on different racks, the noise significantly increases on Daint. One of the reasons is adaptive routing, that can sometimes unnecessarily select longer paths [21].

On-premise HPC systems are characterized by latency noise with higher intensity, due to the generally lower latency (cf. Sec. III). As a consequence, fluctuations in the latency have a larger relative impact (but a smaller absolute impact). It is worth remarking that, regardless of the absolute impact of noise on the latency, as we will show in Sec. V, noise at scale can negatively impact the performance, even on systems characterized by a lower base latency.

Figure 10 shows the latency noise for different instance types for the four cloud providers (with instances running in the same rack). We observe that on GCP normal instances are characterized by higher noise compared to HPC instances, whereas, on Azure, HPC instances experience a few high-intensity noise events. We did not observe significant differences between the different instance types on AWS, whereas on Oracle we observe a higher noise on HPC instances compared to normal instances.

Observation 6: Latency noise affects both cloud and on-premise HPC systems, and can increase the latency by more than 100x (in a single case up to 35000x). Except

that for GCP, HPC instances are not characterized by a lower latency noise than normal instances.

C. Bandwidth Noise

To measure bandwidth noise, we run large message ping-pongs between two nodes for one hour, and record each sample. For each provider, we select the size of the message that saturates the bandwidth, as well as the optimal number of connections. Even in this case we did not observe significant intra-day variability.

We report the bandwidth noise for HPC instances (bare-metal when applicable) for different node distances in Figure 11. We normalize the bandwidth with respect to the maximum and, to improve the readability of the plot, we only plot the bottom 0.1% samples (i.e., those with the lowest bandwidth). For completeness, we report in Table II the maximum and average bandwidth. We observe that, when the two nodes are on the same rack, AWS, GCP, and Oracle are the most affected by bandwidth noise. On Oracle, bandwidth noise is clustered in time, causing persistent drops that last for up to 0.6 seconds.

Daint and DEEP-EST are almost unaffected by bandwidth noise, whereas Azure and Alps experiences some bandwidth drops. When the two nodes are on different racks, almost all the systems (except for DEEP-EST and Azure) experience an

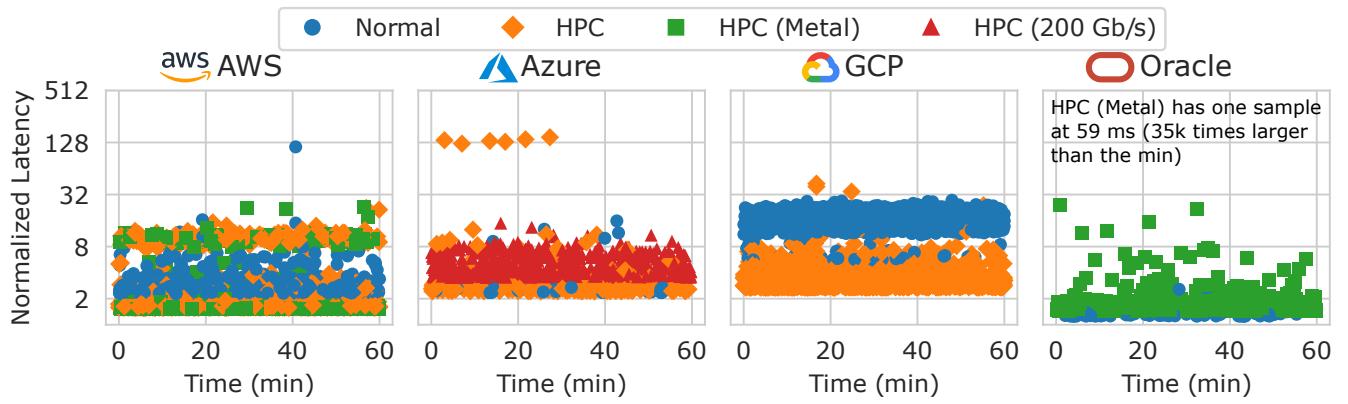


Fig. 10. Latency noise for different instance types. Base latency is reported in Table II.

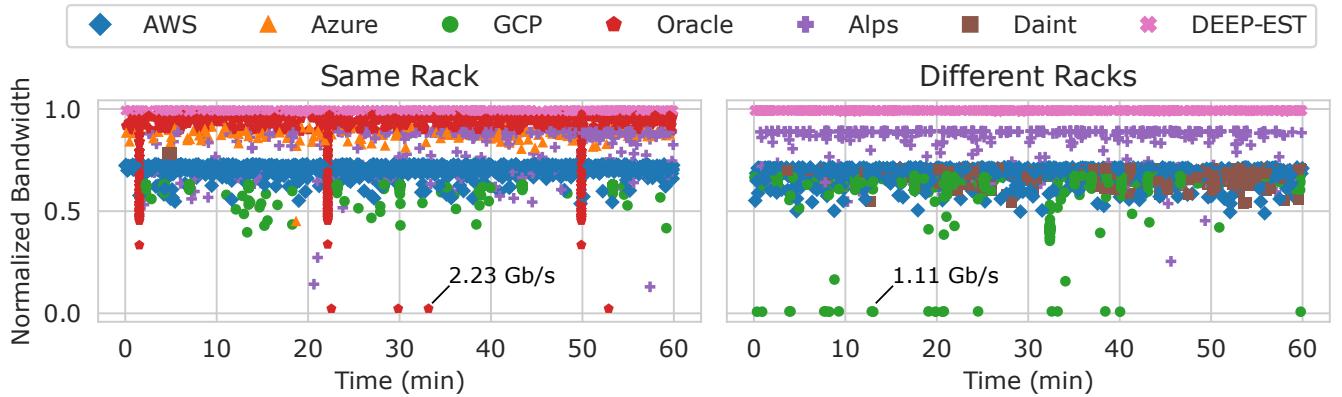


Fig. 11. Bandwidth noise for different node distances for 100 Gb/s HPC instances. Base bandwidth is reported in Table II.

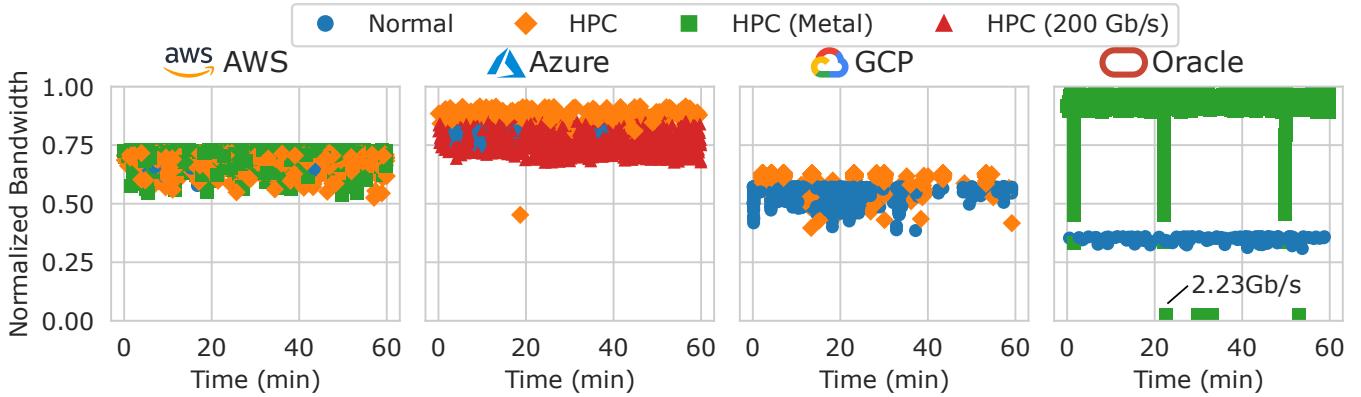


Fig. 12. Bandwidth noise for different instance types. Base bandwidth is reported in Table II.

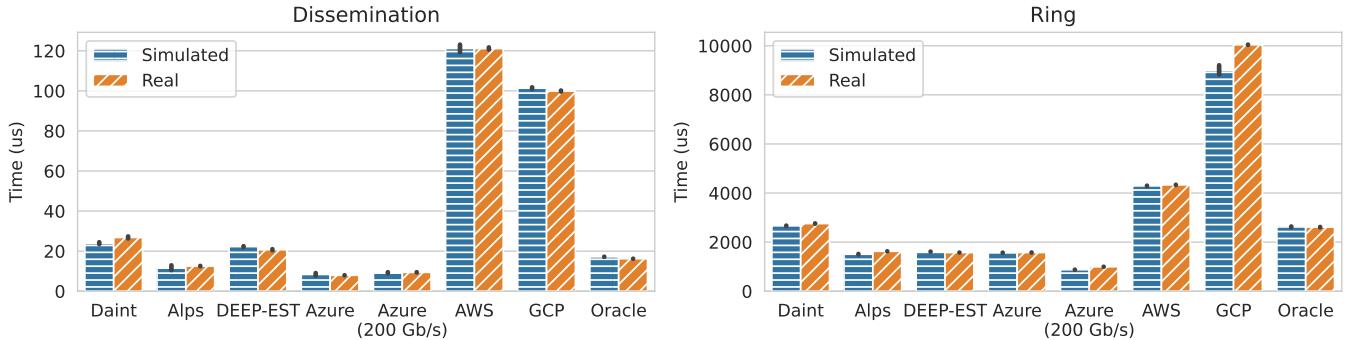


Fig. 13. Comparison between measured and simulated times (on HPC instances) for 16B dissemination and 16MiB ring collectives on 16 nodes. Vertical lines at the top of the boxes represent the 95% confidence interval.

increase in bandwidth noise, with GCP having a few samples experiencing a severe bandwidth drop.

Figure 12 shows the bandwidth noise for the different instance types, when the two nodes are on the same rack. We do not observe significant differences between the different instance types on AWS and GCP. On Azure, we observe a slightly higher bandwidth noise on 200 Gb/s instances, for the same reason discussed for the latency noise (small congestion events have a larger relative effect at higher bandwidth). On Oracle, normal instances are severely affected by bandwidth noise, with many samples characterized by a bandwidth equal to 40% of the maximum achievable bandwidth.

Observation 7: Cloud systems are more affected by bandwidth noise than on-premise systems. When nodes are on two different racks, the impact of bandwidth noise increases on GCP and Daint. On GCP and Oracle, normal instances are more affected by bandwidth noise.

V. LARGE-SCALE SIMULATIONS

To assess the impact of noise at scale, we simulate the performance of collective operations with the LogGOPSim simulator [44], [45]. This allows us to analyze the impact of different types of noise in a controlled environment and to estimate the scalability of distributed applications at scale.

A. Simulation Setup and Validation

LogGOPSim uses the LogGOPSim network model to simulate the execution of parallel algorithms and entire applications. The simulator takes as input the LogGOPSim parameters and a program to simulate on an arbitrary number of nodes. The program can either be specified through the *Group Operation Assembly Language* (GOAL [68]) or through MPI traces. GOAL specifications are composed of a series of send and receive operations for each process, plus synthetic computational operations, and dependencies among these operations. The simulator can also take as an input an OS noise trace, and we extended it to use latency and bandwidth noise traces. Every time a message is sent, instead of simulating a fixed

term for the latency or the bandwidth, a value is drawn from the latency and bandwidth noise distributions measured on the real system. Those distributions are built according to the measurement taken with Netgauge (see Sec. IV).

Before simulating the impact of noise at a large scale, we validate the simulator on different collective algorithms. We implemented a program that generates the MPI code of a specific distributed algorithm starting from its GOAL specification. We then execute the generated code on 16 nodes (due to quota limitations, this is the largest allocation that we could get on all the cloud providers at the time of writing.) on the HPC instances. We show in Figure 13 the time measured on the real systems and the time simulated by LogGOPSim starting from the corresponding GOAL trace.

We consider two different widely-used collective operations: a dissemination algorithm (like those used in barriers and small allreduce operations) and a ring-based collective (like the one used in reduce-scatter, allgather, and allreduce operations on large data). Dissemination is executed with 16B messages, and ring with 16MiB messages. We observe how the simulated time closely match the measured time, with a relative error lower than 10% on both collectives. Similar results have been observed on 4, 8, and 32 nodes, and shown in Appendix A. We also run the validation up to 128 nodes on Daint (not shown in the plot), observing a relative error below 5%.

B. Analysis of Noise on Dissemination Collectives

To analyze the impact of different types of noise on the scalability of parallel applications, we simulated the performance of a 16B dissemination collective on the different systems, and we report the results in Figure 14. For the cloud providers, we only consider HPC instances. We report the results without any noise, with only OS noise or only network noise, and with both noise types. For AWS, we do not report the results for HPC non bare-metal instances, because we did not observe any difference from HPC bare-metal.

We use different scales for the Y-axes in each subplot for readability purposes. We repeated each experiment 1000 times, and we report the entire distribution of the samples as boxplots. In each boxplot, the middle line represents the

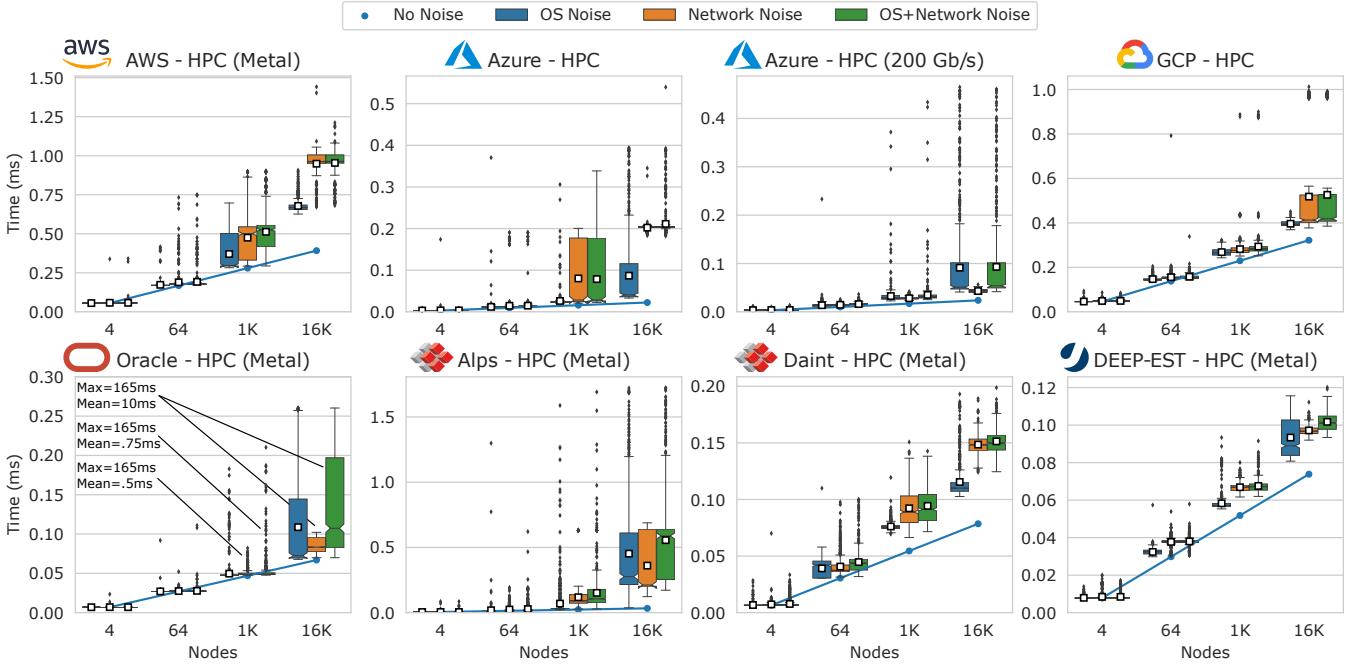


Fig. 14. Simulation of the scalability of a 16B dissemination algorithm, with and without OS and network noise. Y-axes have different scales.

median, the notch around the median is the 95% confidence interval, and the small white square indicates the mean. The upper and lower limits of the box indicate the first quartile (Q_1) and the third quartile (Q_3). Being $IQR = Q_3 - Q_1$ the interquartile range, the lower and upper whiskers indicate the smallest sample $> Q_1 - 1.5 \cdot IQR$, and the largest sample $< Q_3 + 1.5 \cdot IQR$, respectively. Samples outside the whiskers are outliers and reported as black diamonds.

We first focus on the performance in the ideal case where noise does not affect the systems. We observe that, even without noise, both AWS and GCP are characterized by a lower performance than Azure, Oracle, and the on-premise systems, up to an order of magnitude lower. This is due to the latency on AWS and GCP being one order of magnitude larger than that of the other systems (see Table II). Starting from 1K nodes AWS performance is severely affected by noise. As highlighted in Sec. IV-A, AWS systems are those experiencing OS noise with the highest intensity, and this significantly increases the runtime of the collective operation, up to 2x on 16K nodes. Even without any OS noise, network noise would still increase the runtime by 50%. We did not observe amplification effects when adding both types of noise.

Something similar occurs on Azure HPC instances, where network noise increases the average runtime by more than a factor of 4 on 16K nodes, with a few outliers increasing the runtime by 10 times compared to a noiseless execution. On the other hand, 200Gb/s instances are mostly affected by OS noise since, as shown in Sec. IV-A, latency noise has lower intensity compared to 100Gb/s instances. GCP is characterized by network noise with lower intensity, reflecting on a lower relative increase in the runtime.

On Oracle, we observe a small increase in the median runtime. However, a few outliers (not shown in the plot for the sake of readability) experienced a runtime of 165ms on 16K nodes, increasing the average runtime to 10ms. These outliers are caused by some rare noise events with very high intensity (as discussed in Sec. IV-B). OS and network noise severely affect on-premise systems as well, with Daint experiencing a 2x increase in the runtime on 16K nodes due to OS noise, and up to almost 4x due to network noise.

Observation 8: OS and network noise can increase the median runtime up to 2x on 1K nodes, and up to 10x on 16K nodes, both on on-premise and cloud systems.

C. Analysis of Noise on Ring Collectives

We also analyze the impact of noise on bandwidth-dominated ring collectives. This type of collectives are widely used in the training of deep learning models to average gradients [69]. In a ring collective each node iteratively receives a chunk of data from its left neighbor, and sends another chunk of the same size to the right neighbour. An allreduce performed as a ring collective is bandwidth optimal, and particularly effective on a smaller node count and for large data. However, due to the long chain of dependencies (each node has to wait to receive data from the previous node before sending), it is enough to slow down a single message transmission to delay the entire operation.

For this reason, we show in Figure 15 the results of the simulation of a 512MiB ring allreduce collective. First, for all the systems we observe no impact of OS noise on the runtime.

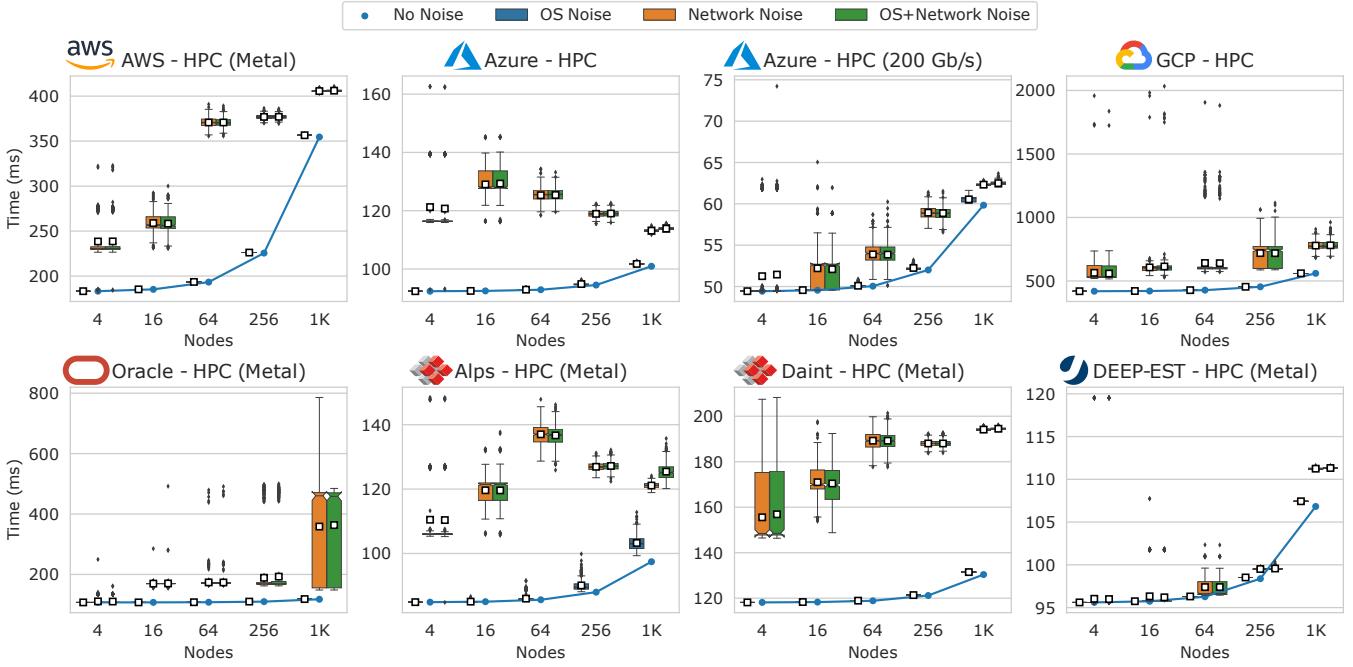


Fig. 15. Simulation of the scalability of a 512MiB ring allreduce collective on HPC instances.

Also, we observe how network (bandwidth) noise impacts the performance also on when running the application on 4 nodes only. On almost all the providers we observe network noise increasing the runtime by almost 50% on 4 nodes. On GCP, a few outliers increase the runtime up to 5 times on 16 nodes, whereas on Oracle the noise increases the average runtime by 4x on 1024 nodes.

Observation 9: Bandwidth noise can severely affect both on-premise and cloud systems, increasing the runtime by 50% even when running at small scale (4 nodes).

D. Analysis of Noise Impact on Cost

Last, we analyze the cost of running HPC workloads in the cloud, by also factoring in the costs due to the different CPUs used by the different providers. We simulate a strawman application running a 128×128 double-precision matrix-matrix multiplication followed by a dissemination collective. The message size in the dissemination phase is 128 KiB, equal to the size of the matrix ($128 \times 128 \times 8$ bytes). This communication pattern represents applications that perform some computation followed by reductions (e.g., deep learning training workloads). We measured the time required to perform matrix multiplications on the different CPUs deployed in the analyzed systems by running a benchmark [70] using MKL on Intel processors, and BLIS on AMD processors.

It is worth remarking that the purpose of this benchmark is not to extensively evaluate CPUs performance (that is outside the scope of this paper), but rather to factor in our cost estimation the difference in costs due to different CPU technology.

Also, collecting traces of real applications and then simulating them would complicate the interpretability of the results. For example, differences between systems due to differences in OS, libraries compilers, and tools would be amplified on complex applications. By using a mockup application, we can instead keep these differences at a minimum and spotlight differences due to the network rather than on other factors.

We report in Figure 16 the results of this analysis on 100 Gb/s instances for different node count, by showing the relative increase in the monetary cost caused by noise over a noiseless execution, due to the increase in the runtime. We considered the on-demand cost, and we only report the results for those systems where the cost is publicly available (see Table I). First, we observe that even when running on 64 nodes, the increase in the cost due to noise ranges from 10 to 50%.

This is exacerbated when increasing the number of nodes, and we observe up to a 2x increase in the cost at 16K nodes. We observe that OS noise only contributes for a fraction of the cost increase, and that much of the increase is caused by network noise. On Oracle, we observe the highest cost increase due to network noise, due to some outliers with very high latency, as discussed in Sec. IV-B and Sec. V-B. Also, it is worth remarking that, a larger cost increase does not necessarily mean a larger cost and that, despite the noise, Azure and Daint are still more cost effective than AWS and GCP for this type of workload.

The increase in the cost due to noise also depends on the ratio between communication and computation time. To further analyze this, we repeat the same experiment by simulating matrix multiplication of larger matrices (8192×8192), followed by a 512 MiB allreduce (implemented with a ring

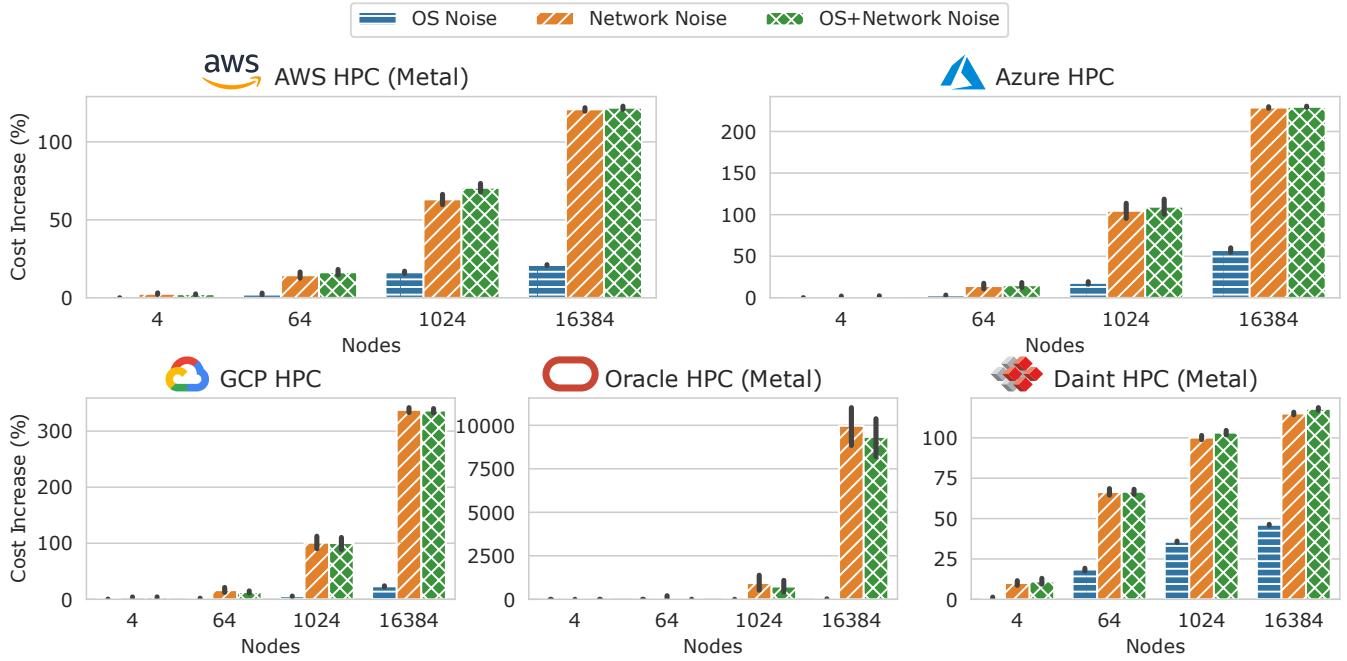


Fig. 16. Simulation of the cost 128×128 double-precision matrix multiplications followed by a 128KiB dissemination, on different node count. Black vertical lines at the top of the boxes represent the 95% confidence interval.

collective). While in the previous experiment the communication accounted for more than 90% of the overall execution time, in this case it accounts for less than 20% of the overall time. We show the results of this analysis in Figure 17.

For this workload, OS noise does not play a significant role, since the application is not latency sensitive. On the other hand, network noise increases the monetary cost of at least 5% on all the systems. On GCP, it causes an additional 15% cost on 4 nodes, and up to 30% on 256 nodes.

Observation 10: Network noise can have a significant impact on the monetary cost of running a distributed application. This is even true at small scale, and for applications that are not dominated by communication.

VI. DISCUSSION

A. Recommendations

In general, we observe some differences between the network performance of 100 Gb/s HPC networks of cloud and on-premise HPC systems. For example, AWS and GCP cannot saturate the bandwidth with a single connection. Accordingly, HPC applications running on those two systems should perform, when possible, multiple concurrent communications at any given time to fully exploit the underlying HPC network. Also, both AWS and GCP are characterized by a higher latency than on-premise systems, and are thus not particularly suited for latency-sensitive HPC applications. On the other hand, both Azure and Oracle show latency and bandwidth much similar to those of on-premise HPC systems.

In general, OS and network noise affect in similar ways the performance of both cloud and on-premise HPC systems. However, there are some exceptions. For example, on Oracle we observed some recurrent and severe bandwidth drops (and latency spikes). This also happened on GCP, but only when the two VMs were allocated on different racks. To mitigate the impact of bandwidth noise on GCP, users should take some extra care when creating their VMs, trying to minimize bandwidth-heavy communications between VMs on different racks. On Daint, communications between nodes on different racks also experience latency noise with high intensity. This is partially due to the routing algorithm, and users could use existing techniques to mitigate this issue [21].

Through simulations we observed that, whereas OS noise has a limited impact on performance, network noise can reduce the performance up to 2x, both at small and large scales. Unfortunately, there is no way for the users to mitigate this, but we want we raised awareness on the impact this issue might have on the performance and cost of running applications (both on cloud and on-premise HPC systems).

B. Limitations

Our study has some limitations that must be considered to have a complete picture. For example, network noise might depend on which and how many applications were concurrently ran by other users on the systems. Because this is not under our control, we ran each experiment multiple times and at different times of the day, with the aim of capturing different behaviours. However, we observed a consistent behavior. Both Alps and Daint were usually fully utilized, as we observed

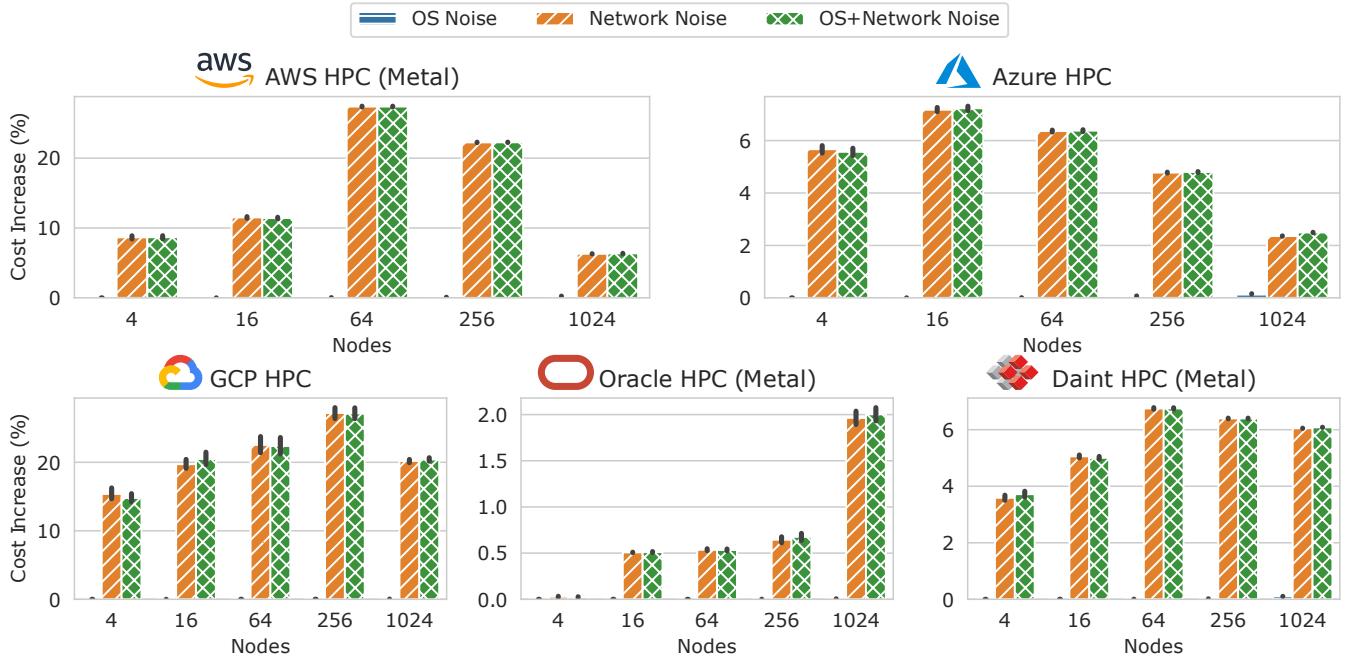


Fig. 17. Simulation of the cost 8192×8192 double-precision matrix multiplications followed by a 512 MiB ring allreduce, on different node count. Vertical lines at the top of the boxes represent the 95% confidence interval.

from Slurm (the job scheduler deployed on those two systems). DEEP-EST was usually quiet, while we could not get any information about how many applications were running at any given time on the cloud systems.

Similarly, on the cloud we can only control whether the two VMs are on the same or different racks (and not even on all the providers). However, the specific placement and the distance between the VMs would also have a different impact on performance and noise that, however, we are not able to control on such a fine granularity.

Also, it is in general not possible to precisely separate network noise from OS noise. Indeed, part of the network communication is executed on the host and is thus influenced by OS noise. After discussing with Cray HPE engineers, we found out that this is likely one of the reasons why, for example, we observed a high latency noise on Alps. This issue has been fixed through updates to the OS, but those updates were not deployed on Alps before the publication of this paper.

In this work, we used the LogGOPSim simulator, that we extended to simulate latency and bandwidth noise, and instantiated with the parameters we measured on the different systems. Due to limitations in the number of instances we could create on cloud systems, we could validate our simulator only on up to 32 nodes. Although network performance might change at larger scales (e.g., due to a higher average distance between compute nodes, and to higher path diversity), we still believe that these simulations help understanding the general performance trend, and to simulate and isolate the impact of OS and network noise.

VII. RELATED WORK

A. Cloud Benchmarking

To improve network performance, recently cloud providers introduced homogeneous HPC instances and low-latency networks with RDMA support. Therefore, various works asked if this could be a game-changer for tightly coupled applications. Sadooghi et al. [71] compare AWS with a private cloud by considering a wide variety of metrics including memory hierarchy, compute, network, and I/O performance, as well as the cost normalized to memory capacity and bandwidth, and to compute performance. Other metrics include job queueing and turnaround time [29]. In this work, instead, we focus on network performance and noise, and on its impact on scalability at scale, by performing our analysis on four different cloud providers and three on-premise HPC systems.

Mohammadi and Bazhirov [72] use Linpack benchmarks to compare different cloud computing vendors and a traditional supercomputer. Their results show that Microsoft Azure provides the best results thanks to the low latency Infini-band interconnect network that facilitates efficient scaling, and performance on the public cloud can be comparable to modern traditional supercomputing systems. Aljamal et al. [73] benchmark various Azure Cloud instances with the NASA parallel benchmark, highlighting the benefits of having high bandwidth capable networking for global communication patterns.

Guidi et al. [12] evaluate benchmarks and applications on AWS, showing that the cloud may have higher bandwidth and lower latency than on-prem systems, especially for medium-

large sized messages. Fernandez [74] evaluates single instance and cluster performance on five cloud providers using microbenchmarks to measure performance of collective MPI operations and the HPCG benchmark. Results show that only 100 Gb/s instances exhibits good scalability.

While these works mainly focus on benchmarking of compute performance and application scalability, all of them highlight the importance of the network in achieving profitability. To the best of our knowledge, no other works assess the impact of network noise on cloud. In this paper we provide an in-depth assessment of network performance and noise, and simulate its impact on small collectives at scale. Our simulation methodology can also be applied on other workloads, to understand if and how they can benefit from running on the cloud.

Most of the above works analyze cloud providers using well-known network benchmarks, such as OSU micro-benchmarks [75] or Intel MPI benchmarks [76]. These tools however only provide aggregated measurements over multiple runs, that would then hide noise effects. In this work we rely on the Netgauge measurement tool [60], that provides detailed per-sample measurements, as well as tools for estimating LogGP parameters and OS noise.

B. Network Noise

Various works investigate the impact of network noise [25], [24], [77]. Chunduri et al. [22] analyze different sources of performance variability, and the impact of the routing algorithm on collective operations. Priscari et al. [78] propose job allocation strategies to minimize the contention on the links. Smith et al. [23] study the impact of network noise on both dragonfly and fat-tree networks and propose an adaptive routing algorithm for fat-trees. De Sensi et al. [21] propose an application-aware routing algorithm to mitigate network noise and improve application performance on dragonfly networks. However, all these works focus on on-prem systems and, to the best of our knowledge, this is the first systematic analysis of network noise on HPC cloud systems and on its impact on scalability.

VIII. CONCLUSIONS

The HPC community often looks at cloud systems with skepticism, amongst the concerns about their costs, and the suspicion about their ability to efficiently run large-scale tightly coupled applications. In this work, we analyze network performance and noise, and their impact on the scalability of large-scale computations.

Like on-prem systems, cloud interconnection networks suffer from network performance variability. We used small scale performance and noise measurements to assess the impact of noise at a larger scale through simulations. After validating the simulation environment, we showed that all the providers are affected by OS and network noise both at small and large scale, and for different communication patterns. Those effects must be taken into account when we look at the cost profitability of such systems.

Although cloud systems are updated frequently, we believe the methodology we used (benchmark at a small scale and simulation at a larger scale) can be utilized on any large scale system to evaluate the impact of OS and network noise at scale. This approach can be a valuable tool for administrators and architects to assess the extent of OS and network noise. Also, it can help researchers, institutions, and businesses that want to quickly evaluate the potential impact of noise on the scalability of a cloud based solution for their HPC workloads without paying the cost of renting (and getting access to) a large number of machines.

ACKNOWLEDGMENT

We would like to thank for their support, feedback, and insightful discussions: CSCS; Duncan Roweth from Cray HPE; Evan Burness from Microsoft Azure; Matt Koop and Brendan Bouffler from AWS; Bill Magro, David Wetherall, Jiuxing Liu, and Rick Jones from GCP; Patrick Saltzmann, Anupam Karmakar, and Calebe Kuenzle from Oracle Cloud. We also thank our shepherd, Kevin Vermeulen, and all anonymous reviewers for their insightful feedback and suggestions, which substantially improved the content and presentation of this paper.

This work has been partially funded by the European Research Council (ERC)  grant PSAP (grant no. 101002047), the European Union's Horizon Europe programme projects DEEP-SEA (grant no. 955606), and RED-SEA (grant no. 955776). Daniele De Sensi was supported by an ETH Postdoctoral Fellowship (19-2 FEL-50).

REFERENCES

- [1] D. Reed, D. Gannon, and J. Dongarra, “Reinventing high performance computing: Challenges and opportunities,” <https://arxiv.org/abs/2203.02544>, 2022.
- [2] M. Parashar, M. AbdelBaky, I. Rodero, and A. Devarakonda, “Cloud paradigms and practices for computational and data-enabled science and engineering,” *Comput. Sci. Eng.*, vol. 15, no. 4, pp. 10–18, 2013. [Online]. Available: <https://doi.org/10.1109/MCSE.2013.49>
- [3] C. Peña-Monferrer, R. Manson-Sawko, and V. Elisseev, “Hpc-cloud native framework for concurrent simulation, analysis and visualization of cfd workflows,” *Future Generation Computer Systems*, vol. 123, pp. 14–23, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X21001291>
- [4] K. Saurabh, S. Adavani, K. Tan, M. Ishii, B. Gao, A. Krishnamurthy, H. Sundar, and B. Ganapathysubramanian, “Case study of sarscov-2 transmission risk assessment in indoor environments using cloud computing resources,” in *2021 SC Workshops Supplementary Proceedings (SCWS)*. Los Alamitos, CA, USA: IEEE Computer Society, nov 2021, pp. 79–86. [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/SCWS55283.2021.00020>
- [5] M. Ohue, K. Aoyama, and Y. Akiyama, “High-performance cloud computing for exhaustive protein–protein docking,” in *Advances in Parallel & Distributed Processing, and Applications*, H. R. Arabnia, L. Deligiannidis, M. R. Grimaldi, D. D. Hodson, K. Joe, M. Sekijima, and F. G. Tinetti, Eds. Cham: Springer International Publishing, 2021, pp. 737–746.
- [6] V. Navale and P. E. Bourne, “Cloud computing applications for biomedical science: A perspective,” *PLOS Computational Biology*, vol. 14, no. 6, pp. 1–14, 06 2018. [Online]. Available: <https://doi.org/10.1371/journal.pcbi.1006144>
- [7] AWS, “Aws graviton processor,” <https://aws.amazon.com/it/ec2/graviton/>, 2022, accessed: 31-Mar-2022.

- [8] Amazon, “Amazon ec2 p4d instances – highest performance for ml training and hpc applications in the cloud,” <https://aws.amazon.com/ec2/instance-types/p4/>, 2022, accessed: 31-Mar-2022.
- [9] Google, “Cloud tpu,” <https://cloud.google.com/tpu>, 2022, accessed: 31-Mar-2022.
- [10] Azure, “Fpga optimized virtual machine sizes,” <https://docs.microsoft.com/en-us/azure/virtual-machines/sizes-field-programmable-gate-arrays>, 2022, accessed: 31-Mar-2022.
- [11] Y. Zhai, M. Liu, J. Zhai, X. Ma, and W. Chen, “Cloud versus in-house cluster: Evaluating amazon cluster compute instances for running mpi applications,” in *State of the Practice Reports*, ser. SC ’11. New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: <https://doi.org/10.1145/2063348.2063363>
- [12] G. Guidi, M. Ellis, A. Buluç, K. Yelick, and D. Culler, *10 Years Later: Cloud Computing is Closing the Performance Gap*. New York, NY, USA: Association for Computing Machinery, 2021, p. 41–48. [Online]. Available: <https://doi.org/10.1145/3447545.3451183>
- [13] D. Thaler and C. Hopps, “Rfc2991: Multipath issues in unicast and multicast next-hop selection,” USA, 2000.
- [14] L. Shalev, H. Ayoub, N. Bshara, and E. Sabbag, “A cloud-optimized transport protocol for elastic and scalable hpc,” *IEEE Micro*, vol. 40, no. 6, pp. 67–73, 2020.
- [15] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagalal, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hözle, S. Stuart, and A. Vahdat, “Jupiter rising: A decade of clos topologies and centralized control in google’s datacenter network,” in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM ’15. New York, NY, USA: Association for Computing Machinery, 2015, p. 183–197. [Online]. Available: <https://doi.org/10.1145/2785956.2787508>
- [16] J. Cao, R. Xia, P. Yang, C. Guo, G. Lu, L. Yuan, Y. Zheng, H. Wu, Y. Xiong, and D. Maltz, “Per-packet load-balanced, low-latency routing for clos-based data center networks,” in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT ’13. New York, NY, USA: Association for Computing Machinery, 2013, p. 49–60. [Online]. Available: <https://doi.org/10.1145/2535372.2535375>
- [17] M. Alizadeh, T. Edsall, S. Dhamapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese, “Conga: Distributed congestion-aware load balancing for datacenters,” in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM ’14. New York, NY, USA: Association for Computing Machinery, 2014, p. 503–514. [Online]. Available: <https://doi.org/10.1145/2619239.2626316>
- [18] S. Ghorbani, Z. Yang, P. B. Godfrey, Y. Ganjali, and A. Firoozshahian, “Drill: Micro load balancing for low-latency data center networks,” in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 225–238. [Online]. Available: <https://doi.org/10.1145/3098822.3098839>
- [19] D. De Sensi, S. Di Girolamo, K. H. McMahon, D. Roweth, and T. Hoeffer, “An in-depth analysis of the slingshot interconnect,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC ’20. IEEE Press, 2020.
- [20] C. B. Stunkel, R. L. Graham, G. Shainer, M. Kagan, S. S. Sharkawi, B. Rosenberg, and G. A. Chochia, “The high-speed networks of the summit and sierra supercomputers,” *IBM Journal of Research and Development*, vol. 64, no. 3/4, pp. 3:1–3:10, 2020.
- [21] D. De Sensi, S. Di Girolamo, and T. Hoeffer, “Mitigating network noise on dragonfly networks through application-aware routing,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC ’19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: <https://doi.org/10.1145/3295500.3356196>
- [22] S. Chunduri, K. Harms, S. Parker, V. Morozov, S. Oshin, N. Cherukuri, and K. Kumaran, “Run-to-run variability on xeon phi based cray xc systems,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC ’17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: <https://doi.org/10.1145/3126908.3126926>
- [23] S. A. Smith, C. E. Cromey, D. K. Lowenthal, J. Domke, N. Jain, J. J. Thiagarajan, and A. Bhatele, “Mitigating inter-job interference using adaptive flow-aware routing,” in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2018, pp. 346–360.
- [24] X. Yang, J. Jenkins, M. Mubarak, R. B. Ross, and Z. Lan, “Watch out for the bully! job interference study on dragonfly network,” in *SC ’16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2016, pp. 750–760.
- [25] T. Hoeffer, T. Schneider, and A. Lumsdaine, “The Effect of Network Noise on Large-Scale Collective Communications,” *Parallel Processing Letters (PPL)*, vol. 19, no. 4, pp. 573–593, Aug. 2009.
- [26] Y. A. Liu, X. L. Liu, F. N. Li, H. Fu, Y. Yang, J. Song, P. Zhao, Z. Wang, D. Peng, H. Chen, C. Guo, H. Huang, W. Wu, and D. Chen, “Closing the “quantum supremacy” gap: Achieving real-time simulation of a random quantum circuit using a new sunway supercomputer,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC ’21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: <https://doi.org/10.1145/3458817.3487399>
- [27] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, “Performance analysis of high performance computing applications on the amazon web services cloud,” in *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, 2010, pp. 159–168.
- [28] S. Coghlan and K. Yelick, “The magellan final report on cloud computing,” 12 2011. [Online]. Available: <https://www.osti.gov/biblio/1076794>
- [29] A. Marathe, R. Harris, D. K. Lowenthal, B. R. de Supinski, B. Rountree, M. Schulz, and X. Yuan, “A comparative study of high-performance computing on the cloud,” in *Proceedings of the 22nd International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC ’13. New York, NY, USA: Association for Computing Machinery, 2013, p. 239–250. [Online]. Available: <https://doi.org/10.1145/2493123.2462919>
- [30] S. Chang, R. Hood, H. Jin, S. Heistand, J. Chang, S. Cheung, J. Djomehri, G. Jost, and D. Kokron, “Evaluating the Suitability of Commercial Clouds for NASA’s High Performance Computing Applications: A Trade Study,” *NASA NAS Technical Report*, vol. 1, no. May, pp. 1–46, 2018.
- [31] M. A. S. Netto, R. N. Calheiros, E. R. Rodrigues, R. L. F. Cunha, and R. Buyya, “Hpc cloud for scientific and business applications: Taxonomy, vision, and research challenges,” *ACM Comput. Surv.*, vol. 51, no. 1, jan 2018. [Online]. Available: <https://doi.org/10.1145/3150224>
- [32] T. 500, “Top 500 list,” <https://www.top500.org/>, 2022, accessed: 31-Mar-2022.
- [33] K. Bergman, “Empowering flexible and scalable high performance architectures with embedded photonics,” in *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2018, pp. 378–378.
- [34] G. Michelogiannakis, Y. Shen, M. Y. Teh, X. Meng, B. Aivazi, T. Groves, J. Shalf, M. Glick, M. Ghobadi, L. Dennison, and K. Bergman, “Bandwidth steering in hpc using silicon nanophotonics,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC ’19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: <https://doi.org/10.1145/3295500.3356145>
- [35] J. L. Cox, “Evolution of Optical Technologies in the Cloud Infrastructure,” <https://www.youtube.com/watch?v=r3GJt7AiGuc>, 2022, accessed: 31-Mar-2022.
- [36] Microsoft, “High performance computing VM sizes,” <https://docs.microsoft.com/en-us/azure/virtual-machines/sizes-hpc>, 2022, accessed: 31-Mar-2022.
- [37] ———, “Adaptive Routing on Azure HPC ,” <https://techcommunity.microsoft.com/5/azure-compute-blog/adaptive-routing-on-azure-hpc/ba-p/1205217>, 2022, accessed: 11-Mar-2022.
- [38] M. Dalton, D. Schultz, J. Adriaens, A. Arefin, A. Gupta, B. Fahs, D. Rubinstein, E. C. Zermenio, E. Rubow, J. A. Docauer, J. Alpert, J. Ai, J. Olson, K. DeCaboer, M. De Kruijf, N. Hua, N. Lewis, N. Kasinadhuni, R. Crepaldi, S. Krishnan, S. Venkata, Y. Richter, U. Naik, and A. Vahdat, “Andromeda: Performance, isolation, and velocity at scale in cloud network virtualization,” in *Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI’18. USA: USENIX Association, 2018, p. 373–387.

- [39] Forbes, “Oracle Cloud Had A Banner 2021 And Is Very Credible,” <https://www.forbes.com/sites/patrickmoorhead/2022/02/08/oracle-cloud-had-a-banner-2021-and-now-very-credible/>, 2022, accessed: 30-Jun-2022.
- [40] A. Froidmont, “Running Applications on Oracle Cloud Using Cluster Networking,” <https://blogs.oracle.com/cloud-infrastructure/post/running-applications-on-oracle-cloud-using-cluster-networking>, 2022, accessed: 30-Jun-2022.
- [41] S. N. S. Centre, “Pricing: Pay-As-You-Go Service,” <https://2go.cscs.ch/offering/pricing/>, 2022, accessed: 02-May-2022.
- [42] B. Alverson, E. Froese, L. Kaplan, and D. Roweth, “Cray xc series network,” *Cray Inc. White Paper WP-Aries01-1112*, 2012.
- [43] E. Suarez, A. Kreuzer, N. Eicker, and T. Lippert, *The DEEP-EST project*, ser. Schriften des Forschungszentrums Jülich IAS Series. Jülich: Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag, 2021, vol. 48, pp. 9–25. [Online]. Available: <https://juser.fz-juelich.de/record/905812>
- [44] T. Hoefer, T. Schneider, and A. Lumsdaine, “LogGOPSIM - Simulating Large-Scale Applications in the LogGOPS Model,” in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, Jun. 2010, pp. 597–604.
- [45] SPCL, “LogGOPSIM simulator,” <https://github.com/spcl/LogGOPSIM>, 2022, accessed: 31-Mar-2022.
- [46] Amazon, “Amazon AWS,” <https://aws.amazon.com/>, 2022, accessed: 11-Mar-2022.
- [47] Google, “Google GCP,” <https://cloud.google.com/>, 2022, accessed: 11-Mar-2022.
- [48] Microsoft, “Microsoft Azure,” <https://azure.microsoft.com/>, 2022, accessed: 11-Mar-2022.
- [49] Oracle, “Oracle Cloud Infrastructure,” <https://www.oracle.com/cloud/>, 2022, accessed: 11-Mar-2022.
- [50] Swiss National Supercomputing Centre, “Piz Daint Supercomputer” <https://www.cscs.ch/computers/piz-daint/>, 2022, accessed: 11-Mar-2022.
- [51] ——, “Alps Supercomputer,” <https://www.cscs.ch/computers/alps/>, 2022, accessed: 06-May-2022.
- [52] Amazon, “Aws parallelcluster user guide,” <https://docs.aws.amazon.com/parallelcluster/latest/ug/cluster-definition.html#base-os>, 2022, accessed: 31-Mar-2022.
- [53] Microsoft, “Centos-hpc vm images,” <https://docs.microsoft.com/en-us/azure/virtual-machines/workloads/hpc/configure#centos-hpc-vm-images>, 2022, accessed: 31-Mar-2022.
- [54] Google, “Creating an hpc-ready vm instance,” <https://cloud.google.com/compute/docs/instances/create-hpc-vm>, 2022, accessed: 31-Mar-2022.
- [55] Jülich Supercomputing Centre, “DEEP-EST System Overview,” https://deeptrac.zam.kfa-juelich.de:8443/trac/wiki/Public/User_Guide/System_overview, 2022, accessed: 06-May-2022.
- [56] P. MacArthur, Q. Liu, R. D. Russell, F. Mizerow, M. Veeraraghavan, and J. M. Dennis, “An integrated tutorial on infiniband, verbs, and mpi,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2894–2926, 2017.
- [57] Amazon, “Aws nitro card,” <https://aws.amazon.com/ec2/nitro/>, 2022, accessed: 31-Mar-2022.
- [58] Intel, “Accelerating High-Speed Networking with Intel I/OAT,” <https://www.intel.com/content/dam/doc/white-paper/i-o-acceleration-technology-paper.pdf>, 2022, accessed: 11-Mar-2022.
- [59] G. Cloud, “N2, N2D, C2, and C2D higher bandwidth configuration,” https://cloud.google.com/compute/all-pricing#high_bandwidth_configuration, 2022, accessed: 02-May-2022.
- [60] T. Hoefer, T. Mehlhan, A. Lumsdaine, and W. Rehm, “Netgauge: A Network Performance Measurement Framework,” in *Proceedings of High Performance Computing and Communications, HPCC’07*, vol. 4782. Springer, Sep. 2007, pp. 659–671.
- [61] Amazon, “Amazon EC2 instance network bandwidth,” <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-network-bandwidth.html>, 2022, accessed: 11-Mar-2022.
- [62] OpenMPI, “FAQ: Tuning the run-time characteristics of MPI TCP communications,” <https://www.open-mpi.org/faq/?category=tcp#tcp-multi-links>, 2022, accessed: 11-Mar-2022.
- [63] Microsoft, “Fsv2 series,” <https://docs.microsoft.com/en-us/azure/virtual-machines/fsv2-series>, 2022, accessed: 31-Mar-2022.
- [64] K. B. Ferreira, P. Bridges, and R. Brightwell, “Characterizing application sensitivity to os interference using kernel-level noise injection,” in *SC ’08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, 2008, pp. 1–12.
- [65] T. Hoefer, T. Schneider, and A. Lumsdaine, “Characterizing the influence of system noise on large-scale applications by simulation,” in *SC ’10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, 2010, pp. 1–11.
- [66] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman, “Loggp: Incorporating long messages into the logp model for parallel computation,” *Journal of Parallel and Distributed Computing*, vol. 44, no. 1, pp. 71–79, 1997. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0743731597913460>
- [67] A. Bhatele, K. Mohror, S. H. Langer, and K. E. Isaacs, “There goes the neighborhood: Performance degradation due to nearby jobs,” in *SC ’13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2013, pp. 1–12.
- [68] T. Hoefer, C. Siebert, and A. Lumsdaine, “Group Operation Assembly Language - A Flexible Way to Express Collective Communication,” in *ICPP-2009 - The 38th International Conference on Parallel Processing*. IEEE, Sep. 2009.
- [69] T. Ben-Nun and T. Hoefer, “Demystifying parallel and distributed deep learning: An in-depth concurrency analysis,” *ACM Comput. Surv.*, vol. 52, no. 4, aug 2019. [Online]. Available: <https://doi.org/10.1145/3320060>
- [70] Intel, “Benchmarking GEMM on Intel Architecture Processors,” <https://www.intel.com/content/www/us/en/developer/articles/technical/benchmarking-gemm-with-intel-mkl-and-blis-on-intel-processors.html>, 2022, accessed: 31-Mar-2022.
- [71] I. Sadooghi, J. H. Martin, T. Li, K. Brandstatter, K. Maheshwari, T. P. P. de Lacerda Ruivo, G. Garzoglio, S. Timm, Y. Zhao, and I. Raicu, “Understanding the performance and potential of cloud computing for scientific applications,” *IEEE Transactions on Cloud Computing*, vol. 5, no. 2, pp. 358–371, 2017.
- [72] M. Mohammadi and T. Bazhirov, “Comparative benchmarking of cloud computing vendors with high performance linpack,” in *Proceedings of the 2nd International Conference on High Performance Compilation, Computing and Communications*, ser. HP3C. New York, NY, USA: Association for Computing Machinery, 2018, p. 1–5. [Online]. Available: <https://doi.org/10.1145/3195612.3195613>
- [73] R. Aljamal, A. El-Mousa, and F. Jubair, “Benchmarking microsoft azure virtual machines for the use of hpc applications,” in *2020 11th International Conference on Information and Communication Systems (ICICS)*, 2020, pp. 382–387.
- [74] A. Fernandez, “Evaluation of the performance of tightly coupled parallel solvers and mpi communications in iaas from the public cloud,” *IEEE Transactions on Cloud Computing*, pp. 1–1, 2021.
- [75] Network-Based Computing Laboratory - Ohio State University, “OSU Microbenchmarks,” <http://mvapich.cse.ohio-state.edu/benchmarks/>, 2022, accessed: 31-Mar-2022.
- [76] Intel, “Intel MPI Benchmarks,” <https://www.intel.com/content/www/us/en/developer/articles/technical/intel-mpi-benchmarks.html>, 2022, accessed: 31-Mar-2022.
- [77] T. Groves, Y. Gu, and N. J. Wright, “Understanding performance variability on the aries dragonfly network,” in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, 2017, pp. 809–813.
- [78] B. Prisacari, G. Rodriguez, P. Heidelberger, D. Chen, C. Minkenberg, and T. Hoefer, “Efficient task placement and routing of nearest neighbor exchanges in dragonfly networks,” in *Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC ’14. New York, NY, USA: Association for Computing Machinery, 2014, p. 129–140. [Online]. Available: <https://doi.org/10.1145/2600212.2600225>

APPENDIX A SIMULATOR VALIDATION

We repeated the validation process described in Sec. V-A on 4 and 8 nodes and, for the instance types where this was possible, also on 32 nodes (not all the providers sufficiently increased our quota limits to enable the creation of a cluster with 32 nodes). We report the results for 4, 8, and 32 nodes in Figure 18, Figure 19, and Figure 20 respectively. Similarly to the results shown in Sec. V-A, we observe that the simulated runtime closely matches the measured one.

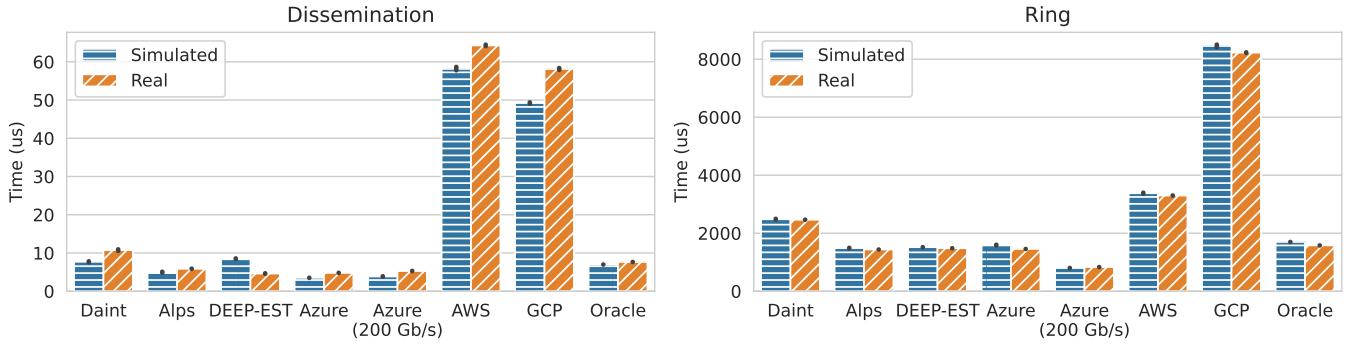


Fig. 18. Comparison between measured (on HPC instances) and simulated times for 16B on dissemination and 16MiB ring collectives on 4 nodes. Black vertical lines at the top of the boxes represent the 95% confidence interval.

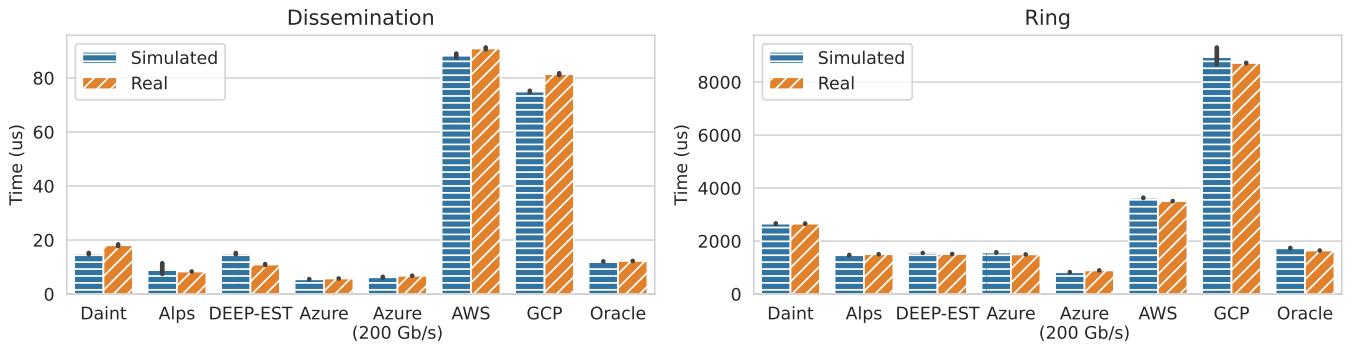


Fig. 19. Comparison between measured (on HPC instances) and simulated times for 16B on dissemination and 16MiB ring collectives on 8 nodes. Black vertical lines at the top of the boxes represent the 95% confidence interval.

APPENDIX B CLUSTERS PROVISIONING TIME

A cluster composed of 2 HPC instances can be provisioned in 22 minutes on Azure, AWS, and Oracle. On the other hand, on GCP this requires less than 1 minute. However, it is worth remarking that on Azure, AWS, and Oracle, the cluster is already provisioned with Slurm and with many software packages often needed for running HPC applications (MPI, Intel MKL, etc...). On the other hand, on GCP everything needs to be installed and configured from scratch and, eventually, the time required to have a fully provisioned cluster is similar on all the four cloud providers.

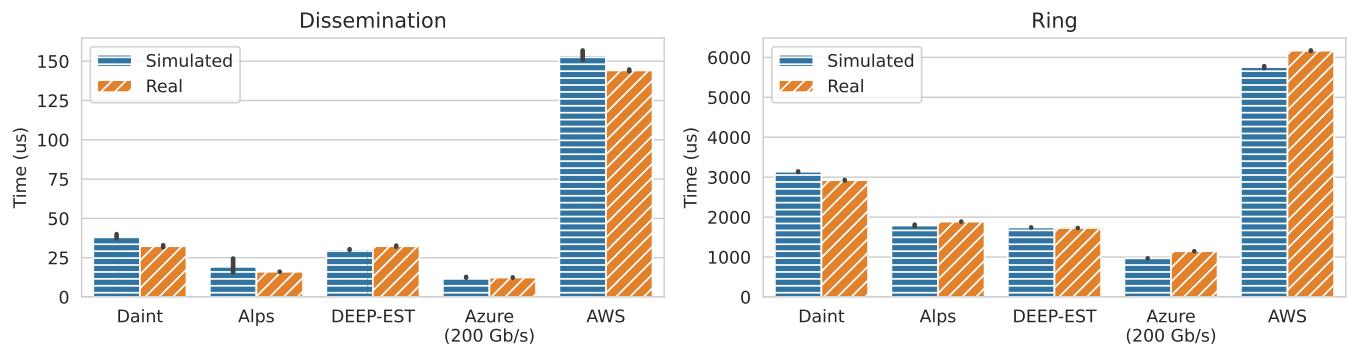


Fig. 20. Comparison between measured (on HPC instances) and simulated times for 16B on dissemination and 16MiB ring collectives on 32 nodes. Black vertical lines at the top of the boxes represent the 95% confidence interval.