Hostping: Diagnosing Intra-host Network Bottlenecks in RDMA Servers

Kefei Liu, *BUPT;* Zhuo Jiang, *ByteDance Inc.;* Jiao Zhang, *BUPT and Purple Mountain Laboratories;* Haoran Wei, *BUPT and ByteDance Inc.;* Xiaolong Zhong, *BUPT;* Lizhuang Tan, *ByteDance Inc.;* Tian Pan and Tao Huang, *BUPT and Purple Mountain Laboratories*

https://www.usenix.org/conference/nsdi23/presentation/liu-kefei

This paper is included in the Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation.

April 17–19, 2023 • Boston, MA, USA

978-1-939133-33-5

Open access to the Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation is sponsored by

Hostping: Diagnosing Intra-host Network Bottlenecks in RDMA Servers

Kefei Liu[†], Zhuo Jiang[§], Jiao Zhang[†‡*], Haoran Wei[†§], Xiaolong Zhong[†], Lizhuang Tan[§], Tian Pan[†‡] and Tao Huang[†‡]

[†]BUPT [‡]Purple Mountain Laboratories
[§]ByteDance Inc.

Abstract

Intra-host networking was considered robust in the RDMA (Remote Direct Memory Access) network and received little attention. However, as the RNIC (RDMA NIC) line rate increases rapidly to multi-hundred gigabits, the intra-host network becomes a potential performance bottleneck for network applications. Intra-host network bottlenecks may result in degraded intra-host bandwidth and increased intra-host latency, which can severely impact network performance. However, when intra-host bottlenecks occur, they can hardly be noticed due to the lack of a monitoring system. Furthermore, existing bottleneck diagnosis mechanisms fail to diagnose intra-host bottlenecks efficiently. In this paper, we analyze the symptom of intra-host bottlenecks based on our long-term troubleshooting experience and propose Hostping, *the first bottleneck monitoring and diagnosis system dedicated to intra-host networks*. The core idea of Hostping is conducting loopback tests between RNICs and endpoints within the host to measure intra-host latency and bandwidth. Hostping not only discovers intra-host bottlenecks we already knew but also reveals six bottlenecks we did not notice before.

1 Introduction

RDMA has been applied to many applications [14] [17] [28] [30] [42] [46] in data centers to achieve high throughput and ultra-low latency. As the last hop of network communication, intra-host networking can significantly impact the performance of network applications. However, the intra-host network is far from flawless, and intra-host bandwidth may degrade due to sudden link failures or occupation by other traffic. Previously, the intra-host bandwidth was much greater than the RNIC line rate (e.g., ~63 Gb/s PCIe Gen 3 x8 for 25 Gb/s RNIC), providing sufficient bandwidth redundancy for RNIC traffic. Therefore, the intra-host network rarely became an obstacle to network communication, and bottlenecks in the host network received little attention.

However, bottlenecks in the host network are on the rise. With the increasing demand for high throughput and ultra-low latency, the RNIC line rate increases rapidly (from 25 Gb/s to 200 Gb/s). In contrast, the intra-host bandwidth does not improve equally (e.g., PCIe bandwidth increases from ~63 Gb/s to ~252 Gb/s). As a result, when intra-host bandwidth degrades, traffic on the RNIC is more likely to be throttled. What is worse, both the topology and traffic patterns within the host become much more complicated, making bandwidth degradation caused by sudden link failures or traffic contention happens more frequently. Besides, as intra-host services become more complex, configuration items in the host also increase considerably, leading to a high probability of misconfigurations. Some of them, such as enabling Access Control Service, will redirect GDR (GPU Direct RDMA) traffic to the CPU, leading to a drastic increase in intra-host latency and severe degradation of intra-host bandwidth.

Intra-host bottlenecks [1] may significantly degrade network performance. In our distributed machine learning system, one single intra-host bottleneck can significantly degrade the whole system and may even block the training process. This phenomenon is common in our data center. When it occurs, operators may need hours to days to diagnose the root cause.

Why do intra-host bottlenecks have such a severe impact? If the intra-host bandwidth is lower than the RNIC receiving rate, the RNIC receive buffer may accumulate or even be saturated. When this occurs in a lossy environment (without PFC) [39], RNIC may drop packets. Since RDMA is vulnerable to packet drops, even a low drop rate will result in drastic throughput degradation [24]. While in a lossless environment, RNIC will send PFC pause frames (Tx pause frames) to the upstream switch's egress port to stop its traffic. If the RNIC sends pause frames continually, it may eventually lead to a PFC storm [21] [24] [36], which may bring down the whole network.

The first two authors contributed equally to this paper. This work is done while Kefei Liu, Haoran Wei, and Xiaolong Zhong are doing a joint research project at ByteDance. (*Jiao Zhang is the corresponding author.)

[1] In the following, we use "intra-host bottleneck" as the bottleneck in the host network and "network bottleneck" as the bottleneck in the inter-host network, i.e., switches and cables.

Therefore, when an intra-host bottleneck occurs, it should be discovered, diagnosed, and resolved as soon as possible.

However, due to the lack of an efficient intra-host bottleneck monitoring system, bottlenecks can hardly be noticed when they occur. When customers complain to the network team about performance degradation, the upper layer service usually has been severely influenced by the bottleneck. In addition, the phenomena caused by intra-host and network bottlenecks may be similar. Thus, when network performance degrades, operators need first to judge whether the host or the network should be blamed. Furthermore, when finding the bottleneck lies in the host, operators need to log in to the host, execute a series of test cases and conduct some profiling tools to infer the bottleneck. The whole process is time-consuming. What is worse, existing profiling tools could only be used for specific devices, such as Intel PCM [2] for Intel CPUs, AMD uProf [1] for AMD CPUs, and Nvidia SMI [9] for Nvidia GPUs. As each host may have devices from different vendors, operators may need different toolsets for each diagnosis, which brings additional learning and execution overhead.

To solve the limitations above, we propose Hostping, *the first bottleneck monitoring and diagnosing system dedicated to intra-host networks*. It could be deployed on all RDMA servers with low overhead and adapt to devices from different vendors. When intra-host bottlenecks occur, Hostping could quickly discover them and automatically diagnose their root causes. Thus, when network performance degrades, we can rapidly judge whether the host or the network should be blamed.

We need to address three challenges to achieve these design targets. Firstly, we need to find and measure metrics that could effectively discover and diagnose intra-host bottlenecks. Secondly, we need to keep responsive to intra-host bottlenecks with low overhead. Finally, we need to efficiently diagnose intra-host bottlenecks based on measured data.

Based on our long-term troubleshooting experience, we realized that leveraging intra-host bandwidth and latency as metrics could effectively discover and diagnose most intra-host bottlenecks. This guides the core idea of Hostping: conduct loopback tests between RNICs and endpoints (GPUs and memory nodes [33]) within the host to measure intra-host latency and bandwidth. By registering memory regions in different endpoints, Hostping could evaluate the latency and bandwidth of any intra-host path that a message received by an RNIC can take. To keep Hostping responsive to intra-host bottlenecks without degrading application performance, we design a hardware monitor to determine when to launch it. Finally, we propose an efficient diagnosing mechanism that could effectively identify the root cause of intra-host bottlenecks even under the interference of service traffic on RNICs.

We evaluate Hostping on over 300 servers in our distributed machine learning system. During the deployment, Hostping not only discovers intra-host bottlenecks we already knew but also reveals six bottlenecks we did not notice before, such as CPU root port failures and memory channel flapping. To summarize, this paper makes the following contributions:

- We analyze the symptom of intra-host network bottlenecks based on our long-term troubleshooting experience and realize that most intra-host bottlenecks have one or both of the following symptoms: intra-host bandwidth degradation and intra-host latency increase.
- We design Hostping, the first bottleneck monitoring and diagnosing system dedicated to intra-host networks.
- We propose an efficient diagnosing mechanism that could effectively identify the root cause of intra-host bottlenecks even under the interference of service traffic on RNICs.

2 Background & Motivation

2.1 Intra-host Bottlenecks

When sending/receiving a message, the RNIC will read/write it from/to an intra-host endpoint (e.g., memory node, GPU) through multi-hops in the host network, such as PCIe links, memory channels, and inter-socket buses (e.g., Intel QPI [51]/UPI [11] and AMD xGMI [12]). We refer to the round-trip latency and the maximum available bandwidth between the RNIC and the endpoint as ***intra-host latency*** and ***intra-host bandwidth***[2], respectively.

Previously, intra-host bandwidth was much greater than the RNIC line rate, providing sufficient bandwidth redundancy. Therefore, the host rarely became an obstacle to network communication, and intra-host bottlenecks received little attention. In recent years, with the increasing demand for high throughput and ultra-low latency from applications, the RNIC line rate has increased rapidly. In contrast, the intra-host bandwidth does not improve equally. As a result, when intra-host bandwidth degrades due to link failures or contention from other intra-host traffic, it is more likely to trigger network performance degradation.

What is worse, both the topology and traffic patterns within the host become much more complex, making the intra-host bandwidth degradation commonplace [13] [16] [19] [35] [37]. To satisfy the ever-increasing demand for computation capability, more GPUs and RNICs are integrated into one single host. For example, the latest Nvidia DGX-A100 [5] server incorporates 8 Nvidia A100 GPUs and 4 Mellanox 200 Gb/s RNICs. This leads to much more complicated intra-host traffic patterns and more bandwidth contention. In addition, as the number of root ports [43] on the CPU socket is limited, more PCIe switches are required to interconnect these devices. As a result, the intra-host topology becomes more complex, leading to more frequent intra-host link failures.

[2]It could be further divided into sending bandwidth from the endpoint to the RNIC and receiving bandwidth from the RNIC to the endpoint. If not explicitly mentioned, it indicates the minimum value of the sending and receiving bandwidth.

Furthermore, as intra-host services become more complicated, configuration items in the host also increase considerably, leading to a high probability of misconfigurations. Among them, some misconfigurations may lead to severe intra-host bottlenecks. For example, ACS (Access Control Service) is a PCIe configuration used in IO virtualization. GDR is a widely used communication method in machine learning, which uses the GPU to communicate directly with the RNIC without any involvement of the CPU and host memory. However, all GDR traffic will be redirected to the CPU with ACS enabled, leading to a drastic increase in intra-host latency and severe degradation of intra-host bandwidth.

2.2 The Impact of Intra-host Bottlenecks

When bottlenecks appear in the host, the intra-host bandwidth may be lower than the RNIC receiving rate, and the RNIC receive buffer may accumulate. If the receive buffer is saturated in a lossy environment (without PFC) [39], the RNIC will drop packets. Since RDMA is vulnerable to packet drops, even a low drop rate will result in drastic throughput degradation [24]. While in a lossless environment, when the RNIC receive buffer exceeds a threshold, it will send pause frames to the upstream switch's egress port to stop its traffic. If the RNIC sends pause frames continually, it may finally lead to a PFC storm, which may bring down the whole network.

One single intra-host bottleneck may significantly degrade the distributed machine learning system. To achieve better training performance, developers aggregate more and more servers in a distributed system. However, this leads to more frequent performance bottlenecks. In data-parallel training, before updating the neural network parameters, all involved GPUs need to aggregate their local gradients [16] [28] [45]. In this process, GPUs may communicate in one or several rings [22] [38] [41] consisting of intra-host links (e.g., NVLinks [8], PCIe links) and network links to achieve optimal bandwidth utilization. This ring-based communication is extremely sensitive to network and intra-host bottlenecks. A single RNIC suffering from degraded intra-host bandwidth may significantly slow down the aggregation process of the whole system. We conducted a ring-based nccl all-reduce test [7] with eight hosts, and each host has a 200 Gb/s RNIC for network communication. Fig.1 shows the throughput of each host during the test. In this scenario, an RNIC's PCIe link has degraded bandwidth due to a link failure, leading to a slow sending/receiving rate. As a result, the throughput for all the hosts drops drastically to 50 Gbps (~70% lower than the ideal).

Frequent intra-host bottlenecks bring more challenges for performance bottleneck diagnosis. When packet drops or bandwidth degradation occur on a path, how to diagnose the root cause? This problem generally lies in the network when few intra-host bottlenecks appear, and operators only need to check each link and switch on the path in sequence. However, as intra-host bottlenecks occur much more frequently,

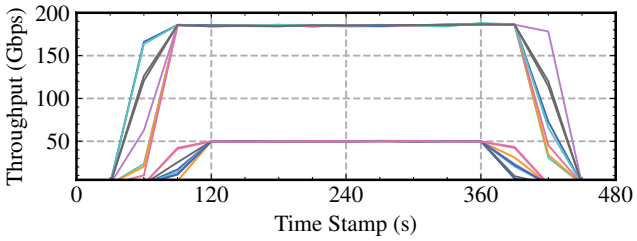

Figure 1: One single bottleneck degrades the throughput of the entire machine learning system by 70%. The upper lines are ideal, and the lower lines are abnormal. The throughput of each host is calculated every 30 seconds.

the same phenomenon may also be caused by the degraded intra-host bandwidth. As a result, operators must first distinguish whether the host or the network should be blamed, which brings more challenges for bottleneck diagnosis.

2.3 Limitations of Existing Intra-host Bottleneck Diagnosis Mechanisms

When an intra-host bottleneck occurs, it must be discovered, diagnosed, and resolved as soon as possible. Unfortunately, as far as we know, there are currently no monitoring and diagnosing systems dedicated to the host network in data centers, and intra-host bottleneck diagnosis is inefficient.

Unresponsive. When bottlenecks occur in a host, they can hardly be noticed in time due to the lack of an efficient intra-host bottleneck monitoring system. However, when customers (e.g., the machine learning team) complain to the network team about performance degradation, the upper layer service has usually been severely influenced. Thus, operators require a responsive monitoring system to quickly discover intra-host bottlenecks, avoiding application performance degradation.

Time-consuming. When a system suffers from degraded performance, operators usually need to run benchmark tests, such as perftest [10] and nccl-test [7], to narrow down the problem. However, these tests reflect "end-to-end" performance, including senders, networks, and receivers. Thus, they cannot quickly determine whether the bottleneck occurs in the network or the host. When finding the bottleneck lies in the host, root cause diagnosis is still challenging due to the complex intra-host topology. Operators need to log in to the host, execute a series of test cases, and conduct some profiling tools to evaluate all intra-host links. The entire process above needs to be conducted manually, which is time-consuming.

Fragmented. When an intra-host link has anomalous performance, operators may need to run some profiling tools to determine whether the link is occupied by other traffic. However, these tools are usually vendor-specific, such as Intel PCM for Intel CPUs, AMD uProf for AMD CPUs, Nvidia SMI for Nvidia GPUs, and Mellanox Neohost [4] for Mellanox RNICs. Unfortunately, each host in data centers may have a different combination of equipment, such as different

network adapters (Mellanox, Broadcom, or Intel), different CPUs (Intel or AMD), and different GPUs (Nvidia or AMD). As a result, when diagnosing bottlenecks in the host, operators need to utilize different combinations of tools, which brings additional learning and execution overhead.

2.4 Targets of Hostping

Considering the limitations above, we desire to develop a dedicated intra-host bottleneck monitoring and diagnosing system, which could be deployed on all RDMA servers with little overhead and adapt to devices from different vendors. When intra-host bottlenecks appear, the system can quickly discover them and automatically diagnose their root causes. Thus, when network performance degrades, we can rapidly judge whether the bottleneck lies in the host or the network. In conclusion, this system should have the following characteristics:

- **Responsiveness:** It should quickly discover intra-host bottlenecks and diagnose their root causes.
- **Deployability:** It should be implementable with commodity hardware.
- **Scalability:** It should be compatible with equipment from different vendors.
- **Lightweight:** It should have negligible interference with services in the host.

3 Hostping Overview

In this section, we will first introduce the challenges we should address to achieve the targets of Hostping (3.1). Then we will analyze the symptoms of intra-host network bottlenecks based on our long-term troubleshooting experience, which guides the core idea of Hostping (3.2). Finally, we will briefly illustrate the framework of Hostping (3.3).

3.1 Challenges

To realize the targets of Hostping, there are three main challenges to be solved:

Find and measure metrics that could effectively discover and diagnose intra-host bottlenecks. As the topology and traffic patterns within the host become much more complex, the root causes of intra-host performance bottlenecks are heterogeneous. We need to find some unified metrics that could effectively uncover intra-host bottlenecks and precisely infer their root causes. Besides, since the intra-host network is like a black box, measuring these metrics with high accuracy is also challenging.

Be responsive to intra-host bottlenecks with low overhead. Diagnosing intra-host performance bottlenecks requires evaluating all the links in the host. Due to the complexity of the host topology, this is not an easy task and will have a non-negligible impact on the applications within the host. For example, active probing consumes CPU memory, GPU video memory, and bus bandwidth. How can we quickly perceive intra-host bottlenecks with low overhead to the performance of applications running in the host?

Effectively diagnose intra-host performance bottlenecks based on measured data. During the operation of Hostping, we will collect many performance data through active probing and monitoring. However, the complex intra-host topology makes it challenging to infer intra-host bottlenecks from scattered data. Besides, the data measured by active probing may be influenced by the service traffic on the RNIC. In this scenario, the degraded performance data does not necessarily mean the emergence of an intra-host bottleneck. We need to find an efficient bottleneck diagnosis mechanism to determine whether there is an intra-host bottleneck and find its root cause effectively based on scattered performance data.

3.2 Symptoms of Intra-host Bottlenecks

As mentioned above, intra-host bottlenecks are varied. How to use the least number of metrics to uncover most intra-host bottlenecks? Based on our long-term troubleshooting experience, we realize that although different root causes may be blamed, most intra-host bottlenecks have one or both of the following symptoms: **intra-host bandwidth degradation** and **intra-host latency increase**. Furthermore, leveraging intra-host bandwidth and latency as metrics could effectively discover and diagnose most intra-host bottlenecks. This guides the core idea of Hostping: conduct loopback tests between RNICs and endpoints within the host to measure intra-host latency and bandwidth. Next, we will introduce these two symptoms and their possible causes.

3.2.1 Bandwidth Degradation

Intra-host bandwidth degrades when an intra-host link is failed or is occupied by other traffic in the host. The RNIC receive buffer will accumulate when the intra-host bandwidth is lower than the RNIC receiving rate. If this situation continues, it will finally trigger packet drops (in lossy environments) or PFC pause frames (in lossless environments), leading to severe network performance degradation.

As the host topology becomes more complicated, the possibility of link failures in the host boosts. In addition, due to the large number of data center hosts, even if link failures are unusual on a particular host, they frequently occur throughout the data center. We encounter abnormal servers even daily in severe cases. What is worse, the locations of failures are varied, requiring a great deal of time for debugging. The host topology inside one of our most used training machines is shown in Fig.2, which has two Intel Xeon CPUs connected through Intel UPI (Intel UltraPath Interconnect). Each CPU

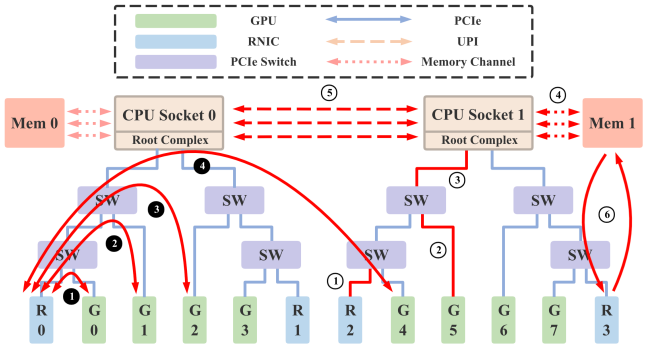

Figure 2: The host topology in one of our most used training machines. ❶-❹ show the GDR distance between an RNIC and a GPU. ①-⑤ show the link failures we encountered in practice, and ⑥ shows intra-host bandwidth degradation due to bandwidth contention.

root complex [43] is attached with four Nvidia A100 GPUs[3] and two Mellanox CX6-DX 200 Gb/s RNICs through multiple PCIe switches. As shown in Fig.2, we have encountered failures of ① RNIC PCIe links, ② GPU PCIe links, ③ CPU root ports, ④ memory channels, and ⑤ UPI in practice. Some issues cannot be detected via static commands such as *lspci* and can only be discovered through benchmark tests.

Furthermore, services in the host are becoming more complicated, leading to more bandwidth contention. When the host bandwidth is occupied by other traffic, traffic on the RNIC may be congested (Fig.2 ⑥). Here, we give two practical examples. First, as RDMA devices are far from flawless, TCP and RDMA traffic may co-exist in the same host to meet high availability and a Service-Level Agreement [21]. However, the processing of TCP in the Linux kernel may consume a lot of memory bandwidth, leading to a slow receiving rate for RDMA traffic. Besides, in the training scenario, a physical machine is usually split into multiple Virtual Machines (VMs) to fully utilize host resources. In this case, communication between two VMs in the same host may trigger loopback traffic, which consumes the RNIC PCIe bandwidth and slows down the receiving rate from other hosts [32]. As shown in Fig.3, both link failures and bandwidth contention may throttle RNIC throughput and trigger a large number of PFC pause frames.

3.2.2 Latency Increase

When sending/receiving a message, the RNIC will read/write it from/to an endpoint (e.g., memory node, GPU) through multi-hops in the host network, such as PCIe links, memory channels, and inter-socket buses. We refer to the round-trip latency from the RNIC receive buffer to the endpoint as *intra-host latency*. Intra-host latency increases when there are too

[3]GPUs are connected via NVLinks and NVSwitches [8] for intra-host GPU-to-GPU communication (not shown in Fig.2).

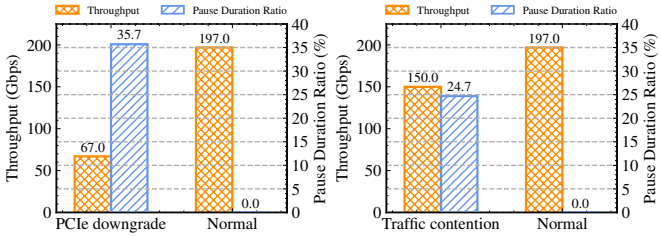

Figure 3: Both link failures (①-⑤) and bandwidth contention (⑥) will lead to intra-host bandwidth degradation, which may throttle RNIC throughput and trigger a large number of PFC pause frames.

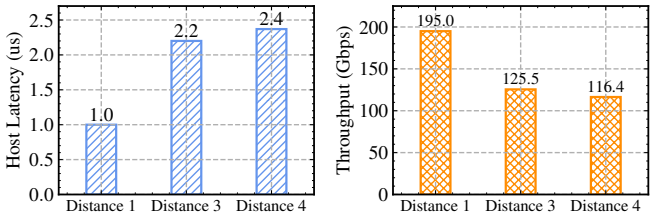

Figure 4: As the GDR read distance increases (from ❶ to ❹), the intra-host latency rises, especially when going through the CPU root complex. Accordingly, the throughput degrades due to limited outstanding read request TLPs.

many hops between the RNIC and the endpoint. High intra-host latency hurts application latency and may significantly degrade intra-host bandwidth. When an RNIC needs to read from an endpoint, it sends PCIe read request TLPs (Transaction Layer Packets) [34] to the endpoint, and the endpoint will respond data to the RNIC after receiving the request. Therefore, when intra-host latency increases, the RNIC needs to send more read requests to sustain the line rate. However, RNICs limit the maximum outstanding read requests. As a result, intra-host bandwidth degrades when intra-host latency increases significantly.

Next, we leverage GDR traffic to illustrate the impact of high intra-host latency on intra-host bandwidth. GDR has been widely used in data centers to improve training performance in distributed machine learning systems. With GDR, the RNIC can write and read GPU video memory directly without using host memory, effectively improving the intra-host latency and intra-host bandwidth. However, GDR suffers from high latency when traffic traverses the CPU root complex. As shown in Fig.2, there are four types of communication distances between an RNIC and a GPU: ❶ traversing a single PCIe switch, ❷ traversing multiple PCIe switches without traversing the CPU root complex, ❸ traversing the CPU root complex without traversing the UPI, and ❹ traversing the UPI.

In the experiment, we use GDR read to test the impact of different communication distances on intra-host latency and bandwidth. We leverage Mellanox Neohost to measure

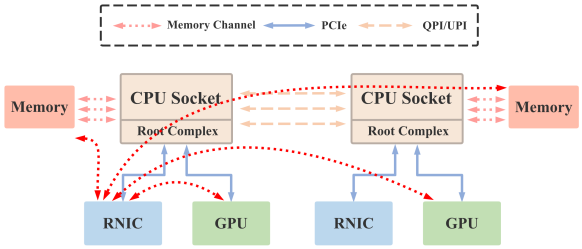

Figure 5: The core idea of Hostping: conduct loopback tests between RNICs and endpoints (GPUs and memory nodes) within the host to measure intra-host latency and bandwidth.

the intra-host latency. Since the latency of ❶ and ❷ is almost the same, we only compare the intra-host latency and the GDR bandwidth of ❶, ❸, and ❹ in the experiment. As shown in Fig.4, when the RNIC communicates with the closest GPU (distance ❶), the host latency is 1 µs, and the RNIC can achieve almost the line rate. For distance ❸, when GDR packets need to pass through the CPU root complex, the host latency rises dramatically to 2.2 µs, and the throughput drops sharply to 125.5 Gbps. As for distance ❹, traversing the UPI bus brings an additional 200 ns delay, and the throughput degrades to 116.4 Gbps. Just 1.4 µs of additional intra-host latency results in a 40% drop in intra-host bandwidth.

3.3 Framework of Hostping

As shown in Fig.5, the core idea of Hostping is conducting loopback tests between RNICs and endpoints within the host to measure intra-host latency and bandwidth and leveraging the measured data to infer intra-host bottlenecks. Hostping is implemented based on commodity RNICs. Thus, it could run on all RDMA servers in data centers.

In the loopback test, the RNIC will read messages from one endpoint to its buffer and then write them back directly. In this process, all communication occurs inside the host without any network participation. Therefore, we could leverage the loopback latency and bandwidth to reflect intra-host latency and bandwidth. Furthermore, by conducting loopback tests between an RNIC and all endpoints in the host, we could evaluate the latency and bandwidth of all intra-host paths that a message received by the RNIC can take. When network performance degrades, if RNICs find no anomalies in loopback tests, we infer that the bottleneck occurs in the network. On the contrary, when the loopback test to an endpoint shows anomalous results, we confirm that a bottleneck exists on the path between the RNIC and the endpoint.

Fig.6 shows the framework of Hostping. The Hostping agent is deployed on RDMA servers and consists of three components: hardware monitor, Hostping engine, and data analyzer. The Hostping engine implements the core logic of Hostping and consists of two functions: (1) leverage RNICs to measure intra-host latency and bandwidth; (2) monitor bus utilization (PCIe links, inter-socket buses, and memory

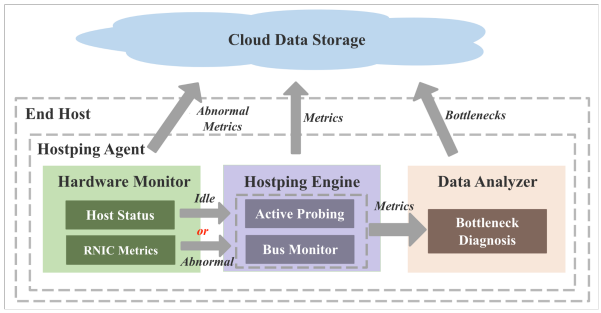

Figure 6: The framework of Hostping.

channels). The hardware monitor judges when to run the Hostping engine based on host status and abnormal metrics on RNICs. The data analyzer is responsible for diagnosing intra-host bottlenecks based on the data collected by the Hostping engine. All these modules will upload the information they collect to the cloud, which will be the basis for subsequent bottleneck diagnosis.

4 Hostping Design

In this section, we will first illustrate the functions of the Hostping engine and how to measure intra-host latency & bandwidth with the loopback test (4.1). Then, we will introduce how to utilize the hardware monitor to keep Hostping highly responsive to intra-host bottlenecks with low overhead (4.2). Finally, we will present how to diagnose intra-host bottlenecks with the data analyzer (4.3).

4.1 Hostping Engine

4.1.1 Measure Intra-host Latency & Bandwidth

Next, we will illustrate how to measure intra-host latency and bandwidth in the Hostping engine. Fig.7 demonstrates the process of the loopback test. First, the Hostping engine leverages `ibv_reg_mr` [3] to register two memory regions (read and write) in an endpoint for sending and receiving, respectively. Next, the Hostping engine uses `ibv_post_send` [3] to post a write WQE (Work Queue Element. Tell the RNIC to read the message of a specified *size* from the read region and write it to the write region) and doorbell the RNIC to fetch the WQE. Then the RNIC will send a request to read the message from the read region. Since the receiver is the same RNIC as the sender, the RNIC will directly write the message back to the write region instead of sending it to the network. Finally, after all PCIe write packets are sent out, the RNIC will generate a completion notification and inform the Hostping engine that the transmission is finished. By measuring the span between the call of `ibv_post_send` and the polling of completion, the Hostping engine could figure out the loopback latency. Moreover, by registering memory regions in different endpoints, we could get the loopback latency between the

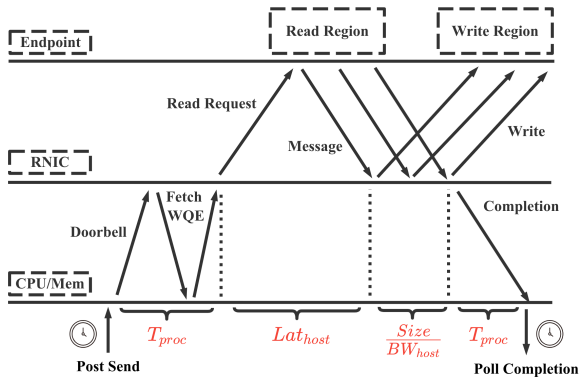

Figure 7: The process of the loopback test. Hostping engine figures out loopback latency by measuring the span between the call of `ibv_post_send` and the polling of completion.

RNIC and any intra-host endpoint. The measured loopback latency could be approximated[4] as follow:

$$Lat = T_{proc} + Lat_{host} + \frac{Size}{BW_{host}} \tag{1}$$

T_{proc} contains two periods: (1) the duration between the call of `ibv_post_send` in the Hostping engine and the RNIC sending the first read request, and (2) the duration between the RNIC sending out all the PCIe write packets and the CPU polling the completion. The intra-host bandwidth (BW_{host}) is determined by the minimum bandwidth in PCIe read and write. Besides, the measured latency also includes intra-host latency (Lat_{host}). When we leverage a small message, the measured latency is close to:

$$\lim_{size \to 0} Lat = T_{proc} + Lat_{host} \tag{2}$$

It is hard to measure T_{proc} on commodity RNICs. Nevertheless, T_{proc} generally remains the same when the RNIC is underutilized and does not suffer from intra-host bottlenecks. In this case, we could leverage the change of small message latency to reflect the variation of intra-host latency. Thus, when the measured latency of a small message increases drastically, we could infer that there is an intra-host bottleneck leading to abnormal intra-host latency. On the contrary, when we use a very large message, the measured latency is close to:

$$\lim_{size \to \infty} Lat = \frac{Size}{BW_{host}} \tag{3}$$

Then the latency reflects intra-host bandwidth. Actually, we do not need to use a very large message in practice. We could use the difference between the latency of large and small messages (Equation 1 - Equation 2) to obtain Equation 3. In practice, our large message size is 128K bytes for 200 Gb/s RNICs, and our small message size is 1 byte.

[4]Here *size* refers to the message size. For simplicity, we do not consider PCIe encapsulation overhead (e.g., TLP header).

4.1.2 Monitor Bus Utilization

While the loopback test could reveal anomalous intra-host paths and links, it fails to diagnose the root cause of anomalies in some scenarios. For example, when the loopback test shows a memory channel has degraded bandwidth, how to further determine whether the root cause lies in traffic contention or a link failure?

To solve this problem, we implement a monitoring module in the Hostping engine to monitor bus utilization (PCIe links, inter-socket buses, and memory channels). Therefore, when the loopback test shows an intra-host link has degraded bandwidth, we could further check its utilization. If the link is overloaded, we infer that the root cause lies in traffic contention. Otherwise, the link is possibly failed. Unlike previous vendor-specific tools, our monitor could automatically adapt to devices from different vendors, and operators no longer need to learn and use various tools for different devices.

4.2 Responsiveness with Low Overhead

When performance bottlenecks occur in the host network, we hope Hostping can automatically, quickly, and accurately locate their root causes. However, high responsiveness and low overhead are usually a trade-off. We can frequently run loopback tests to judge whether there are performance bottlenecks in the host. However, loopback tests consume CPU/GPU memory and intra-host bandwidth, leading to contention with service traffic. Thus, frequent loopback tests will have a non-negligible impact on applications in the host. How could we ensure responsiveness to bottlenecks with low overhead to application performance?

Generally, data center hosts keep switching between busy and idle status. When the host is idle (little traffic on RNICs and all GPUs are inactive), we could frequently run loopback tests to keep responsive to intra-host bottlenecks, regardless of the overhead. When the host is busy with services and the network performance is degraded due to intra-host bottlenecks, abnormal metrics on the RNIC, such as packet drops and Tx pause frames, will usually appear. These metrics are indicators of intra-host bottlenecks. Therefore, we could execute loopback tests when these abnormal metrics appear. This way, Hostping keeps responsive to intra-host bottlenecks with low overhead to application performance.

We implement a hardware monitor in the Hostping agent to achieve the targets above. It (1) monitors host status and abnormal metrics on RNICs and (2) determines when to run the Hostping engine. In general, it has two functions:

- Monitor RNIC throughput and GPU status periodically. If the throughput of all RNICs is less than the threshold Thp_{low}, and all GPUs are idle, execute the Hostping engine to detect if there are intra-host bottlenecks. Otherwise, skip this execution.

- Monitor abnormal metrics on all RNICs. If the Tx pause duration ratio is larger than PFC_{high} or packet drops appear, execute the Hostping engine immediately to diagnose intra-host bottlenecks.

4.3 Bottleneck Analysis

In this section, we will demonstrate how the data analyzer leverages the data collected by the Hostping engine to determine whether there is a bottleneck within the host and diagnose its root cause. We first discuss how to diagnose intra-host bottlenecks when the host is idle. In general, the analyzer determines intra-host path status (normal or abnormal) by comparing the measured intra-host path bandwidth with the baseline and leverages path status to infer anomalous links. This idea is inspired by *binary network tomography* [15] [18] [27]. Besides, the analyzer compares measured intra-host path latency with the baseline to assist in root cause diagnosis. The baseline path bandwidth and path latency are obtained via loopback tests on a batch of idle hosts with the same devices and configurations.

$$y_j = \prod_{i:\ \text{link}_i \in \text{path}_j} x_i, \forall j, \tag{4}$$

The measured path bandwidth reflects the minimum link bandwidth on it. As shown in Equation 4, $y_j, x_i \in \{0,1\}$, represents the status of path j and link i, respectively (1 for normal and 0 for abnormal). If the measured path bandwidth is lower than the baseline by $Abnormal_{th}$, we infer that one or more links on this path suffer from degraded bandwidth and mark this path as abnormal. However, a path with the expected bandwidth is not necessarily bottleneck-free. It depends on whether the RNIC can reach the line rate on this path. For affinitive endpoints of the RNIC (memory nodes[5], GPUs under the same root port as the RNIC), the path bandwidth could reach the RNIC line rate. If the bandwidth of these paths is as expected, we consider them normal. However, as demonstrated in Section 3.2.2, the RNIC could not reach the line rate for GPUs under different CPU root ports due to high intra-host latency. In this case, if the bandwidth of a link degrades but is still higher than the RNIC rate, the measured bandwidth is still close to the baseline. For these paths, we only judge whether they are abnormal based on the baseline.

With adequate path status, we can judge the status of each link within the host. Our algorithm is shown in Algorithm 1. In a symmetric topology like Fig.2, conducting loopback tests between RNICs and their affinitive endpoints could evaluate all intra-host links. Nevertheless, we do full-mesh tests when the host is idle to improve the accuracy of bottleneck inference. If no abnormal paths could be found, we conclude that there is no bandwidth bottleneck. Otherwise, we will diagnose anomalous links based on Algorithm 1. First, we mark all intra-host links as uncertain. Next, we traverse all normal paths and mark all links on them as normal. Then, we traverse all abnormal paths. If an abnormal path has uncertain links, we mark all these links as abnormal, and *abnormal_cnt* records how many RNICs mark a link as abnormal. If all the links on an abnormal path are normal, some links may be flapping. Then we set all the links on this path to gray.

When the host is idle, most bottlenecks could be attributed to *link failures* or *misconfigurations*. The analyzer first judges whether the RNIC is a bottleneck. If the path status between an RNIC and all its affinitive endpoints is abnormal, then the RNIC PCIe link may be failed. If the PCIe link connected to a GPU is marked as abnormal, the analyzer will further check the path latency between the GPU and its affinitive RNIC. If the latency is also abnormal, a misconfiguration (e.g., enabling ACS) may be the root cause. Otherwise, a link failure should be blamed. For other abnormal links, the analyzer diagnoses them as failed links. In addition, links marked as gray in three consecutive loopback tests will be identified as flapping links. All abnormal links and their possible root causes will be reported to operators for further operations, such as hardware inspection and reconfigurations.

When abnormal metrics on an RNIC trigger the Hostping engine, the host is usually busy with services, and some RNICs, especially the abnormal RNIC, may have heavy ser-

Algorithm 1 Detect Links with Bandwidth Degradation

Input: *normal* and *abnormal paths*
Output: *abnormal* and *gray links*

```
1:  function DetectAbnormalLinks()
2:      InitLinkStatus()
3:      for path_j in normal paths do
4:          for link_i in path_j do
5:              link_i.status ← normal
6:      for path_j in abnormal paths do
7:          if ∃ links ∈ path_j in uncertain status then
8:              for link_i in all these links do
9:                  link_i.status ← abnormal
10:                 link_i.abnormal_cnt++
11:         if ∃ links ∈ path_j in abnormal status then
12:             for link_i in all these links do
13:                 if marked abnormal by a new RNIC then
14:                     link_i.abnormal_cnt++
15:         if ∀ links ∈ path_j in normal status then
16:             for link_i in path_j do
17:                 link_i.status ← gray
18:     return abnormal links and gray links
19: function InitLinkStatus()
20:     for link_i in all links do
21:         link_i.status ← uncertain
22:         link_i.abnormal_cnt ← 0
```

[5] We draw this conclusion from the server introduced in Section 3.2.1. For some types of servers, the RNIC cannot reach the line rate when communicating with the memory in remote NUMA nodes.

vice traffic. In this case, the path bandwidth measured by these RNICs will degrade due to the contention of service traffic, even if there is no bottleneck. Thus, we cannot judge the status of these paths according to the bandwidth baseline.

Nevertheless, these RNICs could still indicate abnormal paths. In the server introduced in 3.2.1, applications usually use an RNIC to communicate with its affinitive endpoints (memory nodes, GPUs under the same root port) to achieve optimal performance. Furthermore, memory channels and inter-socket buses generally provide considerable bandwidth redundancy. Therefore, the measured bandwidth between the RNIC and its affinitive endpoints is usually identical if no bottleneck occurs, no matter how much influenced by service traffic on the RNIC. Thus, among the RNIC's affinitive endpoints, if the measured path bandwidth to one endpoint is significantly lower than that to the other endpoints (by $Abnormal_{th}$), we infer this path is abnormal. However, we have no idea whether other paths are normal due to degraded RNIC loopback bandwidth, leading to reduced diagnosis accuracy. As a workaround, we could check whether there are idle RNICs on the host, which could still judge path status according to the baseline.

As applications usually use an RNIC to communicate with its affinitive endpoints, bottlenecks generally occur on the paths between the abnormal RNIC and its affinitive endpoints. Thus, we could focus on finding bottlenecks on these paths. When triggered by abnormal metrics, the Hostping engine only conducts loopback tests between RNICs and their affinitive endpoints. First, this method is sufficient to diagnose the status of the memory channel and the inter-socket bus with low overhead to service traffic. In addition, the service traffic may affect the measured bandwidth between the affinitive GPU of the abnormal RNIC and the RNIC under other root ports. As a result, the analyzer may incorrectly judge these paths as abnormal, leading to an inaccurate diagnosis. Thus, for the links under the same root port as the abnormal RNIC, we only use this abnormal RNIC to judge their status.

The inference of abnormal links is still based on Algorithm 1. However, as RNICs with heavy traffic cannot judge whether a path is normal, some normal links may be marked as abnormal. In this case, links with the highest *abnormal_cnt* are most likely abnormal and should receive more attention. When abnormal metrics trigger the Hostping engine, abnormal links are usually fully loaded. Based on this, we can infer the root cause by monitoring these links. Links with utilization higher than $Util_{high}$ will be diagnosed as *overloaded links*, while *link failures* or *misconfigurations* may be the root cause of other abnormal links. However, as the abnormal RNIC suffers from intra-host bottlenecks, the latency measured by it will rise anomalously. Thus, we cannot judge whether the degraded GPU PCIe link is caused by a link failure or a misconfiguration. Operators then need to do a further inspection. Notably, if no abnormal link could be found, the RNIC PCIe link may be the bottleneck, and the analyzer will further check if it is overloaded with loopback traffic to determine whether traffic contention or a link failure should be blamed.

5 Implementation

For the hardware monitor, throughput and abnormal metrics are provided by our RNIC vendors, and GPU status is obtained based on Nvidia Management Library (NVML) [6]. For the threshold, Thp_{low} is 5% of the RNIC line rate to judge whether the RNIC is idle. PFC_{high} is 3% (every second, transmission is paused by 30ms) to trigger the Hostping engine. During the deployment, the monitor checks the host status every five minutes[6] and collects abnormal metrics every second to decide whether to start the Hostping engine.

For the Hostping engine, we implement the probing module with the verbs API and rdma-core libraries [3]. The bus monitor is implemented based on the API and metrics provided by our vendors: Intel's and AMD's API for CPU root ports, memory channels, and inter-socket buses, NVML for GPU PCIe links, and Mellanox's metrics for RNIC PCIe links.

The data analyzer takes the metrics collected by the Hostping engine as input and infers the most susceptible root causes for intra-host bottlenecks. $Abnormal_{th}$ is 20% to judge whether the latency or bandwidth of a path is abnormal, and $Util_{high}$ is 90% to judge whether a bus is overloaded.

The cloud data storage is implemented based on our time-series database. Every time the Hostping engine starts, all the information collected and deduced by the Hostping agent will be uploaded to the cloud. These data help us better understand the frequency and root causes of bottlenecks. Moreover, operators may need historical data to determine the root causes in some scenarios.

6 Evaluation & Intra-host Bottlenecks Found

We evaluate Hostping on over 300 servers in our distributed machine learning system. The host topology is shown in Fig.2 and introduced in Section 3.2.1, which is the most complex intra-host topology in our data center servers. In this section, we will summarize the bottlenecks we found during the deployment of Hostping. For known bottlenecks, Hostping could effectively diagnose their root causes. In addition, Hostping also reveals six bottlenecks we did not notice before. We roughly classify the bottlenecks found by Hostping into three scenarios according to their root causes.

Scenario 1: Intra-host bandwidth degrades due to link failures. As the host topology becomes more complex, link failures occur frequently. During the deployment, we encountered dozens of instances where failed links resulted in degraded intra-host bandwidth, including failures of *#1 RNIC PCIe links* (Fig.8 (a)), *#2 GPU PCIe links* (Fig.8 (b) & Fig.10

[6] As link failures and misconfigurations infrequently appear in a host, 5 minutes is a fine granularity.

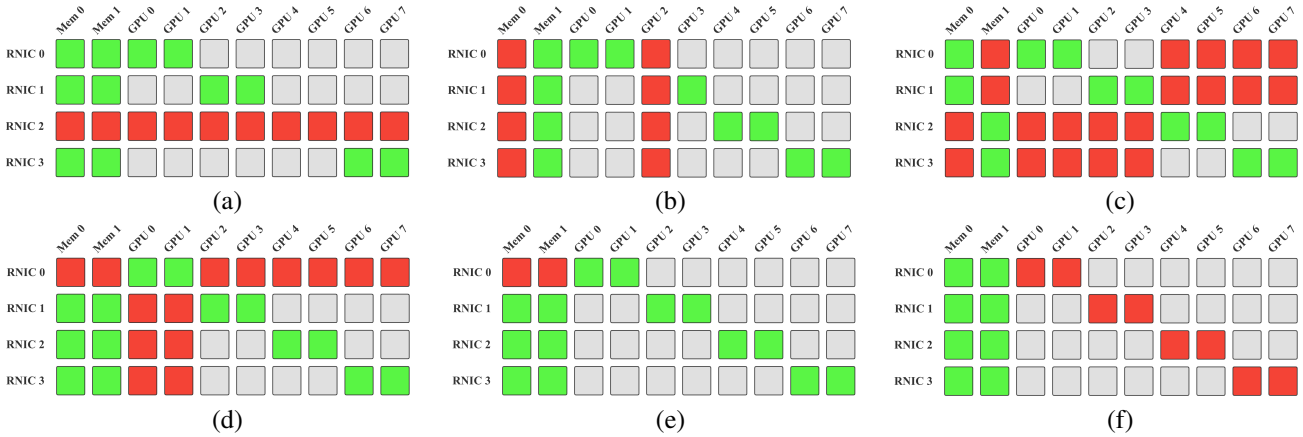

Figure 8: The intra-host end-to-end bandwidth matrices measured by the Hostping engine when hosts are idle. The topology between RNICs and endpoints is shown in Fig.2. (a)-(e) show intra-host bandwidth degradation due to link failures. (f) shows the impact of inappropriate configurations. Green, red, and gray indicate that the path status is normal, abnormal, and uncertain, respectively.

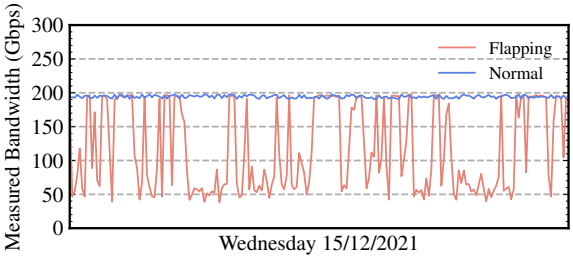

Figure 9: Memory channel flapping on the server. When this occurs, the bandwidth of the memory channel switches between normal and abnormal.

(a)), *#3 memory channels* (Fig.8 (b)), and *#4 UPI* (Fig.8 (c)). For the sake of space, Fig.8 (b) contains #2 and #3. Their problem may be loose PCIe interfaces, dust on connecting fingers, or hardware failures and requires further troubleshooting. Based on the matrix, the analyzer could accurately infer abnormal links. Note that in Fig.10 (a), although RNIC2 has a large amount of service traffic, it could still judge that the path to GPU4 is abnormal according to other measured paths. With Hostping, operators could quickly discover and deal with link failures, avoiding application performance degradation.

[New] *#5 CPU root port failures.* Before deploying Hostping, we only knew four kinds of link failures (#1 to #4). During the deployment, we found that the CPU root port may also experience hardware bandwidth degradation. When this happens, the bandwidth between the RNIC and the GPU under the failed root port is normal. While traffic passes through the failed root port may suffer from degraded bandwidth. The corresponding bandwidth matrix is shown in Fig.8 (d) and (e). They have the same root cause, except that (e) has slight bandwidth degradation, and RNICs under other root ports cannot find anomalies. Nevertheless, Hostping could still accurately diagnose the root cause in this case.

[New] *#6 Memory channel flapping.* With the assistance of Hostping, we found a host suffers from degraded memory channel bandwidth due to a link failure. However, no performance issues could be discovered in subsequent manual testing. By continuously running Hostping and collecting measured data, we found that the root cause lies in the flapping memory channel. As shown in Fig.9, the bandwidth of the host memory channel switches between normal and abnormal. With historical data, we could understand the causes of intra-host bottlenecks more clearly. This case shows the necessity to run Hostping periodically.

Scenario 2: Inappropriate configurations lead to degraded performance. *#7 Enabling ACS results in high PCIe latency.* We have mentioned this case in 2.1. With ACS enabled, all GDR traffic will be guided to the CPU instead of directly to the GPU, resulting in drastic performance degradation. As shown in Fig.8 (f), all PCIe bridges are configured as ACS enabled in this case. As a result, both the latency and bandwidth between the RNIC and the GPU under the same root port turn abnormal. Hostping could accurately diagnose this bottleneck and remind operators to check the configuration.

[New] *#8 Disabling ATS results in high PCIe latency.* We found this case in a virtualized environment. In IO virtualization, if Address Translation Service (ATS) is disabled on the RNIC, all GDR packets will be directed to the CPU root complex for address translation. Similar to #7, with ATS disabled, the latency between the RNIC and the GPU under the same root port increases, leading to drastic bandwidth degradation. By enabling ATS, the translation can be finished in the RNIC to achieve optimal GDR performance.

[New] *#9 Enabling "slow start" on the RNIC.* This is a lossy feature provided by our RNIC vendor. When enabled on an RNIC, the RNIC sending rate will start from a small value instead of the line rate. Although "slow start" could alleviate congestion under Incast scenarios, it increases the completion time of short flows. Thus, it usually remains disabled in most

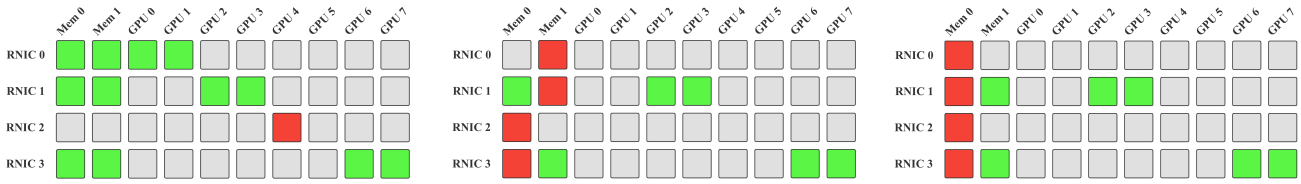

Figure 10: The bandwidth matrices measured when hosts are busy with services. (a) shows hardware bandwidth degradation due to the GPU PCIe link failure. (b) and (c) show degraded intra-host bandwidth caused by traffic contention.

scenarios. When "slow start" is enabled, the RNIC bandwidth to all intra-host endpoints will be lower than the baseline (similar to Fig.8 (a)). At first, Hostping diagnosed the root cause as the RNIC PCIe link failure. However, we found no bottlenecks during continuous bandwidth tests. After configuration inspection, we finally uncovered the root cause.

[New] *#10 Setting "Tx window" too small on the RNIC.* This is also a lossy feature. "Tx window" will limit the maximum in-flight bytes of each queue pair on an RNIC. Therefore, "Tx window" influences the maximum bandwidth a QP could achieve and needs to be set reasonably to alleviate network congestion without degrading throughput. Similar to #9, when an RNIC's "Tx window" is too small, its bandwidth to all intra-host endpoints will be lower than the baseline.

Scenario 3: Intra-host bandwidth degrades due to traffic contention. [New] *#11 Overloaded Inter-socket buses.* During the deployment, we found some malfunctioning applications overloaded the UPI of a host, and cross-socket receiving traffic on RNIC0 triggered a large number of Tx pause frames. Fig.10 (b) shows the corresponding bandwidth matrix measured by Hostping. In this case, RNIC0 and RNIC2 have a large amount of service traffic. Nevertheless, they could still judge that two paths (RNIC0 to mem1 and RNIC2 to mem0) are abnormal according to other measured paths. With the help of the other two idle RNICs, the analyzer infers that the UPI is most likely to be abnormal (with the highest *abnormal_cnt*). Furthermore, leveraging the bus monitor, it diagnoses the root cause as the overloaded UPI. The operator then will find out the traffic source that overloads the UPI.

#12 Overloaded memory channels. TCP and RDMA traffic may co-exist in the same host to keep high availability [21]. In this case, the processing of TCP may consume a lot of memory bandwidth, leading to a slow receiving rate for RDMA traffic. However, we did not discover this case in A100 servers during the deployment of Hostping. As a supplement, we conduct an experiment to evaluate how Hostping behaves when the memory channel is overloaded. We use several processes to overload the channel of mem0. Besides, RNIC0 and RNIC2 receive traffic writing to mem0 and mem1, respectively. Fig.10 (c) shows the corresponding bandwidth matrix. Although RNIC0 and RNIC2 have a large amount of receiving traffic, they could still judge that their paths to mem0 are abnormal. Similar to #11, in this case, the analyzer infers that the channel of mem0 is most likely abnormal and diagnoses the root cause as the overloaded memory channel.

In this scenario, we evaluate the performance of Hostping when intra-host links are overloaded. Furthermore, the results show that Hostping could still effectively diagnose intra-host performance bottlenecks under the interference of service traffic on RNICs.

7 Experiences Learned

Conduct Hostping before running applications. As the intra-host topology becomes more complex, the likelihood of link failures in the host network boosts. Based on our experience, some failures may already exist when the server leaves the factory. Thus, it is essential to evaluate the intra-host network performance before delivering the server to customers. In addition, as intra-host services become more complicated, configuration items in the host also increase considerably, and the configuration methods are varied. For example, the setting of Address Translation Service requires a reboot to take effect. In contrast, Access Control Service is enabled by default and needs to be disabled after each reboot. Furthermore, configurations may not be completed successfully for some reason. Therefore, misconfigurations occur occasionally. We recommend conducting Hostping after each reboot to ensure proper configurations before running applications.

Perceive intra-host bottlenecks with VoQ ECN marking. Although intra-host bottlenecks should be addressed in a targeted manner (e.g., hardware replacement, reconfigurations), we argue that *congestion control mechanisms should be able to perceive intra-host bottlenecks*. Thus, they could alleviate the triggering of packet drops and Tx pause frames to provide better network performance before the bottleneck could finally be resolved. Generally, packets could only be ECN-marked in a switch's egress port, and ECN-based congestion control mechanisms could only perceive network congestion. Fortunately, some latest RNICs provide a new function called VoQ (Virtual Output Queuing) ECN marking, enabling the receiver RNIC to ECN-mark packets when its receive buffer exceeds a threshold. By enabling this function, ECN-based congestion control mechanisms, such as DCQCN [50], can also perceive intra-host bottlenecks. Thus, the sender could timely slow down its sending rate to alleviate the triggering of packet drops or Tx pause frames.

Pay attention to intra-host topologies. Although GDR is

supported at any distance within the host, it is highly recommended not to conduct GDR across CPU root ports. When this occurs, the GDR bandwidth degrades severely, which may lead to a large number of Tx pause frames or packet drops. Furthermore, when testing some AMD servers, we found a large number of Tx pause frames and drastic throughput degradation when the RNIC (200 Gb/s) writes to the memory in the remote socket. This is due to the low cross-socket hardware bandwidth on these servers, and the root cause lies in the host architecture. Thus in these servers, we should avoid using RNICs to communicate with the memory in the remote socket for optimal performance.

8 Related Work

Bottlenecks in the RNIC and intra-host network. With the increasing RNIC line rate, the intra-host network and the RNIC have become potential performance bottlenecks in network communication. Some literature has studied these bottlenecks. Kong et al. [32] implement a tool to help data center operators uncover potential performance bottlenecks in the RNIC. Martinasso et al. [37] analyze congestion behaviors in PCIe fabric and develop a congestion-aware performance model for PCIe communication. Zhang et al. [49] study RDMA sharing characteristics and analyze performance isolation anomalies in RDMA. Neugebauer et al. [40] study the performance impact of PCIe in the host network. Dong et al. [16] analyze different types of traffic congestion in the host network and propose a new server architecture to alleviate intra-host congestion. Faraji et al. [19] show the implication of distance between GPUs on the GPU-to-GPU communication performance in the host network. Farshin et al. [20] study when Intel Data Direct I/O (DDIO) technology becomes a bottleneck in multi-hundred-gigabit networks and how to optimize DDIO-enabled systems for I/O intensive applications. These studies help us better understand the potential bottlenecks in the RNIC and intra-host network.

Bottleneck diagnosis tools. Diagnosis tools could be broadly classified as system-based tools and intra-host tools. System-based tools aim to diagnose performance bottlenecks in the whole system. Pingmesh [25] implements an end-to-end connectivity and latency monitoring system for network troubleshooting and SLA tracking. Netbouncer [44] leverages the IP-in-IP technique to probe designated paths and then diagnoses device and link failures in data center networks. Deepview [48] builds a near-real-time system for virtual disk failure localization. Microscope [23] leverages queuing information at network functions to identify the root causes of performance bottlenecks. SNAP [47] collects network information such as TCP statistics and socket-call logs to pinpoint the problem in data center network applications. In contrast, intra-host tools are dedicated to diagnosing bottlenecks in the host. Haecki et al. [26] implement a latency diagnosis tool to identify the source of network latency in end-host stacks. Mellanox Neohost [4] provides plenty of diagnosis counters on Mellanox RNICs. Nvidia SMI [9] provides the status of Nvidia GPUs. Intel PCM [2] and AMD uProf [1] provide the internal resource utilization of the CPU, including the utilization of buses and interfaces connected to the CPU, such as inter-socket buses, memory channels, and CPU root ports.

9 Conclusion & Future Work

Intra-host networking has become a potential bottleneck for RDMA networks, and intra-host bottlenecks can severely degrade network performance. This paper proposes Hostping to monitor and diagnose intra-host bottlenecks. We analyze the symptom of intra-host bottlenecks based on our long-term troubleshooting experience and realize that most intra-host bottlenecks have one or both of the following symptoms: intra-host bandwidth degradation and intra-host latency increase. Thus, Hostping measures intra-host bandwidth and latency as performance metrics to detect and diagnose intra-host bottlenecks. Furthermore, we propose an efficient diagnosing mechanism that could effectively identify the root cause of intra-host bottlenecks even under the interference of service traffic on RNICs. During the deployment, Hostping not only discovers performance bottlenecks we already knew but also reveals six bottlenecks we did not notice before.

The deployment of Hostping makes us realize that more work needs to be done. Firstly, when the host is busy with services, due to the influence of service traffic, it is challenging to accurately diagnose intra-host link status based on binary path status. If the end-to-end traffic information within the host can be obtained, it will provide more insights into intra-host bottlenecks. Secondly, after finding an overloaded link, we hope Hostping could automatically identify the traffic source, such as malfunctioning applications. Finally, in addition to intra-host network bottlenecks, RNIC bottlenecks, such as scalability problems [29] [30] [31], can also lead to severe network performance degradation. Thus, it is also important to diagnose bottlenecks in the RNIC.

Acknowledgments

We would like to thank our shepherd, Raja Sambasivan, and the anonymous reviewers who helped us improve the quality of this paper. We would also like to thank Huaping Zhou for his insightful feedback. This work is supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61872401 and Grant 62132022, a BUPT-ByteDance Research Project, and the Fok Ying Tung Education Foundation under Grant 171059.

References

[1] AMD uProf. https://developer.amd.com/amd-uprof/.

[2] Intel Performance Counter Monitor. https://github.com/opcm/pcm.

[3] Linux rdma-core. https://github.com/linux-rdma/rdma-core.

[4] Mellanox Neohost. https://support.mellanox.com/s/productdetails/a2v50000000N2OlAAK/mellanox-neohost.

[5] Nvidia DGX-A100. https://www.nvidia.com/en-us/data-center/dgx-a100/.

[6] Nvidia Management Library. https://developer.nvidia.com/nvidia-management-library-nvml.

[7] Nvidia nccl-tests. https://github.com/NVIDIA/nccl-tests.

[8] Nvidia NVLink and NVSwitch. https://www.nvidia.com/en-us/data-center/nvlink/.

[9] Nvidia System Management Interface. https://developer.nvidia.com/nvidia-system-management-interface.

[10] OFED perftest. https://github.com/linux-rdma/perftest.

[11] Intel® Xeon® Scalable Processors Datasheet. https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/2nd-gen-xeon-scalable-datasheet-vol-1.pdf, 2019.

[12] Workload Tuning Guide for AMD EPYC™ 7002 Series Processor Based Servers. https://developer.amd.com/wp-content/resources/56745_0.80.pdf, 2020.

[13] Marcelo Amaral, Jordà Polo, David Carrera, Seetharami Seelam, and Malgorzata Steinder. Topology-Aware GPU Scheduling for Learning Workloads in Cloud Environments. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2017.

[14] Chiranjeeb Buragohain, Knut Magne Risvik, Paul Brett, Miguel Castro, Wonhee Cho, Joshua Cowhig, Nikolas Gloy, Karthik Kalyanaraman, Richendra Khanna, John Pao, et al. A1: A Distributed In-Memory Graph Database. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 329–344, 2020.

[15] Ítalo Cunha, Renata Teixeira, Nick Feamster, and Christophe Diot. Measurement Methods for Fast and Accurate Blackhole Identification with Binary Tomography. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pages 254–266, 2009.

[16] Jianbo Dong, Zheng Cao, Tao Zhang, Jianxi Ye, Shaochuang Wang, Fei Feng, Li Zhao, Xiaoyong Liu, Liuyihan Song, Liwei Peng, et al. EFLOPS: Algorithm and System Co-Design for a High Performance Distributed Training Platform. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 610–622. IEEE, 2020.

[17] Aleksandar Dragojević, Dushyanth Narayanan, Miguel Castro, and Orion Hodson. FaRM: Fast Remote Memory. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 401–414, 2014.

[18] Nick Duffield. Network Tomography of Binary Network Performance Characteristics. *IEEE Transactions on Information Theory*, 52(12):5373–5388, 2006.

[19] Iman Faraji, Seyed H Mirsadeghi, and Ahmad Afsahi. Topology-Aware GPU Selection on Multi-GPU Nodes. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 712–720. IEEE, 2016.

[20] Alireza Farshin, Amir Roozbeh, Gerald Q Maguire Jr, and Dejan Kostić. Reexamining Direct Cache Access to Optimize I/O Intensive Applications for Multi-hundred-gigabit Networks. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 673–689, 2020.

[21] Yixiao Gao, Qiang Li, Lingbo Tang, Yongqing Xi, Pengcheng Zhang, Wenwen Peng, Bo Li, Yaohui Wu, Shaozong Liu, Lei Yan, et al. When Cloud Storage Meets RDMA. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 519–533, 2021.

[22] Andrew Gibiansky. Bringing HPC techniques to deep learning. *Baidu Research, Tech. Rep*, 2017.

[23] Junzhi Gong, Yuliang Li, Bilal Anwer, Aman Shaikh, and Minlan Yu. Microscope: Queue-based Performance Diagnosis for Network Functions. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 390–403, 2020.

[24] Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni, Jianxi Ye, Jitu Padhye, and Marina Lipshteyn. RDMA over Commodity Ethernet at Scale. In *Proceedings of*

the 2016 ACM SIGCOMM Conference, pages 202–215, 2016.

[25] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, Dave Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, and Hua and Chen. Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. *Computer communication review*, 45(4):139–152, 2015.

[26] Roni Haecki, Radhika Niranjan Mysore, Lalith Suresh, Gerd Zellweger, Bo Gan, Timothy Merrifield, Sujata Banerjee, and Timothy Roscoe. How to diagnose nanosecond network latencies in rich end-host stacks. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 861–877, 2022.

[27] Yiyi Huang, Nick Feamster, and Renata Teixeira. Practical Issues with Using Network Tomography for Fault Diagnosis. *ACM SIGCOMM Computer Communication Review*, 38(5):53–58, 2008.

[28] Yimin Jiang, Yibo Zhu, Chang Lan, Bairen Yi, Yong Cui, and Chuanxiong Guo. A Unified Architecture for Accelerating Distributed DNN Training in Heterogeneous GPU/CPU Clusters. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 463–479, 2020.

[29] Anuj Kalia, Michael Kaminsky, and David Andersen. Datacenter RPCs can be General and Fast. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 1–16, 2019.

[30] Anuj Kalia, Michael Kaminsky, and David G Andersen. Using RDMA Efficiently for Key-Value Services. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, pages 295–306, 2014.

[31] Anuj Kalia, Michael Kaminsky, and David G Andersen. FaSST: Fast, Scalable and Simple Distributed Transactions with Two-Sided Datagram RPCs. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 185–201, 2016.

[32] Xinhao Kong, Yibo Zhu, Huaping Zhou, Zhuo Jiang, Jianxi Ye, Chuanxiong Guo, and Danyang Zhuo. Collie: Finding Performance Anomalies in RDMA Subsystems. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 287–305, 2022.

[33] Christoph Lameter. NUMA (Non-Uniform Memory Access): An Overview: NUMA becomes more common because memory controllers get close to execution units on microprocessors. *Queue*, 11(7):40–51, 2013.

[34] Jason Lawley. Understanding Performance of PCI Express Systems. *WP350 (v1. 2). Xilinx*, 97, 2014.

[35] Ang Li, Shuaiwen Leon Song, Jieyang Chen, Jiajia Li, Xu Liu, Nathan R Tallent, and Kevin J Barker. Evaluating Modern GPU Interconnect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect. *IEEE Transactions on Parallel and Distributed Systems*, 31(1):94–110, 2019.

[36] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, et al. HPCC: High Precision Congestion Control. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 44–58. 2019.

[37] Maxime Martinasso, Grzegorz Kwasniewski, Sadaf R Alam, Thomas C Schulthess, and Torsten Hoefler. A PCIe Congestion-Aware Performance Model for Densely Populated Accelerator Servers. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 739–749. IEEE, 2016.

[38] Hiroaki Mikami, Hisahiro Suganuma, Yoshiki Tanaka, Yuichi Kageyama, et al. Massively Distributed SGD: ImageNet/ResNet-50 Training in a Flash. *arXiv preprint arXiv:1811.05233*, 2018.

[39] Radhika Mittal, Alexander Shpiner, Aurojit Panda, Eitan Zahavi, Arvind Krishnamurthy, Sylvia Ratnasamy, and Scott Shenker. Revisiting Network Support for RDMA. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 313–326, 2018.

[40] Rolf Neugebauer, Gianni Antichi, José Fernando Zazo, Yury Audzevich, Sergio López-Buedo, and Andrew W Moore. Understanding PCIe performance for end host networking. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 327–341, 2018.

[41] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799*, 2018.

[42] Jiaxin Shi, Youyang Yao, Rong Chen, Haibo Chen, and Feifei Li. Fast and Concurrent RDF Queries with RDMA-Based Distributed Graph Exploration. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 317–332, 2016.

[43] Richard Solomon. PCI Express Basics. *PCI-SIG, Oct*, 2011.

[44] Cheng Tan, Ze Jin, Chuanxiong Guo, Tianrong Zhang, Haitao Wu, Karl Deng, Dongming Bi, and Dong Xiang. NetBouncer: Active Device and Link Failure Localization in Data Center Networks. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 599–614, 2019.

[45] Xinchen Wan, Hong Zhang, Hao Wang, Shuihai Hu, Junxue Zhang, and Kai Chen. Rat-Resilient Allreduce Tree for Distributed Machine Learning. In *4th Asia-Pacific Workshop on Networking*, pages 52–57, 2020.

[46] Jilong Xue, Youshan Miao, Cheng Chen, Ming Wu, Lintao Zhang, and Lidong Zhou. Fast Distributed Deep Learning over RDMA. In *Proceedings of the Fourteenth EuroSys Conference 2019*, pages 1–14, 2019.

[47] Minlan Yu, Albert Greenberg, Dave Maltz, Jennifer Rexford, Lihua Yuan, Srikanth Kandula, and Changhoon Kim. Profiling Network Performance for Multi-Tier Data Center Applications. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*, 2011.

[48] Qiao Zhang, Guo Yu, Chuanxiong Guo, Yingnong Dang, Nick Swanson, Xinsheng Yang, Randolph Yao, Murali Chintalapati, Arvind Krishnamurthy, and Thomas Anderson. Deepview: Virtual Disk Failure Diagnosis and Pattern Detection for Azure. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 519–532, 2018.

[49] Yiwen Zhang, Yue Tan, Brent Stephens, and Mosharaf Chowdhury. Justitia: Software Multi-Tenancy in Hardware Kernel-Bypass Networks. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 1307–1326, 2022.

[50] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. Congestion Control for Large-Scale RDMA Deployments. *ACM SIGCOMM Computer Communication Review*, 45(4):523–536, 2015.

[51] Dimitrios Ziakas, Allen Baum, Robert A Maddox, and Robert J Safranek. Intel® QuickPath Interconnect Architectural Features Supporting Scalable System Architectures. In *2010 18th IEEE Symposium on High Performance Interconnects*, pages 1–6. IEEE, 2010.