In-Network Monitoring Strategies for HPC Cloud

Conference Paper · February 2024

DOI: 10.1007/978-3-031-57942-4_35

4 authors:

Masoud Hemmatpour
UiT The Arctic University of Norway
27 PUBLICATIONS **135** CITATIONS
SEE PROFILE

Tore H. Larsen
Simula Research Laboratory
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Nikshubha Kumar
BI Norwegian Business School
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Ernst Gunnar Gran
Simula Research Laboratory
43 PUBLICATIONS **416** CITATIONS
SEE PROFILE

In-network monitoring strategies for HPC cloud

Masoud Hemmatpour, Tore Heide Larsen, Nikshubha Kumar, Ernst Gunnar Gran

Abstract

The optimized network architectures and interconnect technologies employed in high-performance cloud computing environments introduce challenges when it comes to developing monitoring solutions that effectively capture relevant network metrics. Moreover, network monitoring often involves capturing and analyzing a large volume of network traffic data. This process can introduce additional overhead and consume system resources, potentially impacting the overall performance of HPC applications. Balancing the need for monitoring with minimal disruption to application performance is a key challenge. In this paper, we study different strategies to enable a low-overhead monitoring system utilizing emerging programmable network devices.

1 Introduction

Cloud service providers strategically design High Performance Computing (HPC) systems with a focus on fault tolerance to guarantee uninterrupted application execution in the face of potential hardware or software failures. Undoubtedly, monitoring constitutes a fundamental element in establishing fault tolerance within HPC systems, affording timely insights into the health, performance, and behaviors of

---

The research project was initiated at SRL and subsequently continued at UiT.

Masoud Hemmatpour
Arctic University of Norway (UiT), e-mail: mhe222@uit.no

Tore Heide Larsen
Simula Research Laboratory (SRL), e-mail: torel@simula.no

Nikshubha Kumar
Norwegian Business School (BI), e-mail: nikshubha.kumar@bi.no

Ernst Gunnar Gran
Norwegian University of Science and Technology (NTNU), e-mail: ernst.g.gran@ntnu.no

diverse components within a HPC cloud infrastructure. The network stands out as a vital constituent within HPC applications. This prominence arises from the prevalent utilization of parallel processing and distributed computing in HPC applications, entailing task division across multiple processors or nodes.

By continuously monitoring network traffic, bandwidth usage, and device performance, administrators can identify areas of congestion, predict future growth requirements, and make informed decisions regarding network upgrades, expansions, or optimization strategies. Furthermore, communication-intensive HPC applications can exploit network monitoring information to improve their performance by efficiently utilizing the network resources [11].

In the early stages of network monitoring, only basic tools like packet sniffers were accessible. These tools enabled network administrators to capture and analyze network traffic, aiding in the identification of connectivity problems and abnormal behavior. Over time, specialized monitoring protocols, such as the Simple Network Management Protocol (SNMP) introduced in the 1990s [5], emerged, specifically tailored to cater to the needs of network monitoring. SNMP enabled the monitoring of network devices and provided a standardized way to collect and manage information about their performance and health. With the advent of Software Defined Networking (SDN) network monitoring has significantly increased due to the easy network observation with a centralized control plane. One of the key benefits of SDN is the centralized control and management of network devices through a software-based controller [9]. This centralized control enables network administrators to have a holistic view of the entire network.

In spite of significant advancements in network monitoring, the task of efficiently monitoring HPC networks remains a challenging endeavor, primarily due to the unique characteristic of HPC network, which necessitates specialized monitoring tools tailored for their requirements. One of the primary factors contributing to this challenge arises from a specific characteristic exhibited by HPC protocols that bypass the processing unit (CPU) and operating system (OS) of a remote system when accessing data, rendering conventional monitoring approaches ineffective in capturing and analyzing network traffic within the HPC context at an end node. Moreover, because of the high cost associated with HPC devices allocating a dedicated processing unit solely for network monitoring is an inefficient and undesirable solution. These circumstances have compelled us to investigate low-overhead approaches for monitoring HPC networks through the utilization of emerging programmable network devices, effectively leveraging their computing power to analyze network traffic.

In Section 2, we present monitoring protocols and existing HPC monitoring tools. In Section 3, we propose and detail our in-network solutions for monitoring HPC networks. Section 4 evaluates the in-network monitoring solution. Finally, in Section 5 we conclude and indicate directions for future work.

2 Network monitoring

The first step in network monitoring is data measurement. Data measurement includes obtaining and preserving data which is then subjected to various analysis for investigation. Measured data can also be visualized, e.g. to provide network operators with easy to interpret network status overviews. Certain control and monitoring protocols, such as the Internet Control Message Protocol (ICMP), are characterized by their simplicity and primarily serve the purpose of reporting communication errors between nodes. Nonetheless, there exist more sophisticated monitoring protocols specifically designed to cater to advanced monitoring requirements. Below, we discuss some of the commonly utilized monitoring protocols:

Simple Network Management Protocol (SNMP) is one of the most used protocols to monitor network status [5]. SNMP for instance can be used to request per-interface port-counters and overall node statistics from a switch. It is implemented in most network devices. Monitoring using SNMP is typically achieved by regularly polling the switch, though switch efficiency may degrade with frequent polling due to CPU overhead. Although vendors are free to implement their own SNMP counters, most switches are limited to counters that aggregate traffic for the whole switch and each of its interfaces, disabling insight into flow level statistics necessary for fine-grained traffic engineering. Therefore, SNMP is not suitable for flow based monitoring.

sFlow is an industry standard network traffic monitoring solution with sampling technology provided by a wide range of network equipment and software application vendors [18]. It is a scalable technique for measuring network traffic, collecting, storing, and analyzing traffic data. One of the advantages of sFlow is scalability of monitoring high speed links without adding significant network load.

Vendor supplied Software Development Kits (SDKs) provide APIs to implement desired functionality in a switch without compromising packet rate performance [14]. In particular, they allow to monitor several objects or performance counters of a network infrastructure. However, these SDKs are vendor specific and limited to the APIs provide by a vendor, thus they do not represent a standard monitoring approach.

2.1 HPC monitoring tools

Communications over traditional LAN technologies and the TCP/IP protocol stack are unfit for HPC networks because of inferior performance characteristics and significant CPU and memory usage. Consequently, a new category of network fabrics emerged, using a technology known as Remote Direct Memory Access (RDMA) [6]. RDMA is a high-performance networking technology that allows direct memory access between computers in a network, bypassing traditional network stack layers and reducing latency. In HPC environments, where data-intensive applications require fast and efficient data transfers, RDMA plays a crucial role. This

makes the monitoring of RDMA protocols more fundamental to ensure optimal performance, to troubleshoot issues, and to maintain the overall health of the network.

Monitoring RDMA networks requires specialized tools and techniques designed to capture and analyze RDMA-specific traffic and performance metrics. These tools provide insights into key parameters such as bandwidth utilization, latency, and congestion [19]. In the following, some of the known tools are discussed:

ibdiagnet [13] is an advanced diagnostic tool designed to analyze and troubleshoot InfiniBand networks. The primary purpose of ibdiagnet is to perform in-depth tests and verification on various components of an InfiniBand network. It can diagnose potential problems related to link connectivity, network congestion, hardware failures, and other issues that may affect the overall performance and reliability of the fabric. ibdiagnet operates by utilizing the Subnet Manager (SM) component of the InfiniBand fabric, which is responsible for managing and configuring the network. It communicates with the SM to obtain information about the network's topology, nodes, and configuration. Based on this data, ibdiagnet performs a series of tests and analyses to evaluate the network's integrity and identify any abnormalities or potential bottlenecks. A significant drawback of utilizing ibdiagnet for network monitoring is the necessity to parse its output in order to extract the desired information. This additional step of parsing introduces complexity and may require specialized tools to effectively extract the relevant data for analysis and interpretation.

rdma_statistic [10] is part of the iproute2 package that provides a collection of utilities for controlling networking and traffic control in Linux. RDMA statistic shows information about counter configuration depending on the option and the command selected. However it is restricted to show information about the RDMA counters of the host machine. In effect, it cannot be extended to obtain information about RDMA communication happening in the network.

Performance Co-Pilot (PcP), developed by Silicon Graphics, ported to Linux from 1999 and eventually open-sourced in the mid-2000s, serves as a framework for supporting performance monitoring and management [16]. It is actively employed by companies such as Netflix and Red Hat to gather high-resolution performance metrics. It is available for all major Linux distributions. Within PCP, the Performance Metrics Collector Daemon (PMCD) acts as an agent running on hosts slated for monitoring. PMCD incorporates various Performance Metrics Domain Agents (PMDA), dynamically loaded libraries with a specified API. Each PMDA collects metrics from a performance metric domain, such as a kernel or a database server. PcP connects to PMCD, requests specific metrics, and PMCD directs the request to the relevant PMDA, subsequently returning the response.

3 HPC In-network monitoring discussion

Effective communication plays a vital role in HPC applications, as a large problem is divided into smaller parts and distributed across multiple machines within

an HPC cluster. Consequently, network utilization and congestion level emerge as critical metrics in HPC networks [1], attracting research attention to monitor and analyze these parameters [2, 7]. In this section, we discuss the available approaches to monitor theses metrics through programmable network devices since there are few studies in this field [4].

Our research initially delved into the SDK monitoring approach, which entails utilizing SDKs provided by vendors for specific Smart Switches and Smart NICs, such as those offered by NVIDIA [14]. These SDKs grant access to hardware counters, enabling the monitoring and analysis of network performance. However, during our investigation, we encountered several limitations and challenges associated with this approach.

First and foremost, relying on an SDK-based monitoring approach is inherently vendor-specific, rendering it inapplicable to network devices from different vendors. This lack of cross-compatibility restricts the generalizability of the monitoring approach. Additionally, certain SDKs, including the NVIDIA Ethernet SDK, were not publicly accessible and necessitated specific circumstances for vendor-granted access. Hence, employing an SDK-based strategy is not the most optimal choice for developing an open and freely accessible cross-platform monitoring system, considering these constraints.

Furthermore, we found that some SDKs were still under development and did not offer complete functionality, despite their documentation promising otherwise. This lack of comprehensive functionality impeded our reliance on these SDKs for thorough monitoring purposes. Acknowledging these limitations, our research shifted its focus towards alternative monitoring approaches that provide more comprehensive and vendor-agnostic solutions for network monitoring. Therefore, we prioritized standard protocols that furnish a structured framework for collecting, analyzing, and visualizing network data. Furthermore, we focus on the RDMA RoCE protocol rather than InfiniBand since RoCE leverages standard Ethernet infrastructures, which is widely deployed in cloud infrastructure and familiar to network administrators.

3.1 Traffic monitoring

Firstly, we examined SNMP in our research. However, our investigation revealed its limitations in monitoring RDMA traffic. Although certain Smart NICs, like BlueField, offer SNMP for monitoring specific counters [12], our evaluation indicated incomplete functionality of SNMP at the time of writing this paper. As a result, we explored alternative standard protocols to address these constraints.

sFlow as a standard monitoring protocol provides real-time visibility and insights into network traffic and performance [18]. It uses packet sampling technology which most of the switches support. It's a sampling mechanism which recently supports RDMA RoCE traffic monitoring which fulfils all the necessary conditions for a HPC network traffic monitoring solution. sFlow has two components: a sFlow agent,

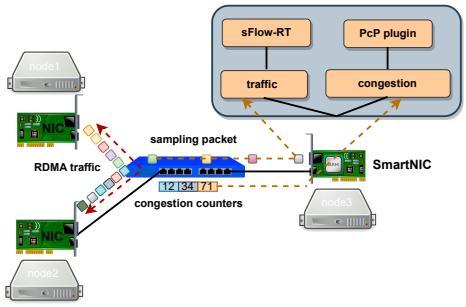

Fig. 1: Testbed consists of three main components to collect and visualize the observed RDMA RoCE traffic.

which is a software process, needs to be enabled within the switch, and a collector runs on a host that collects sFlow datagrams. The packet sampling is typically performed by the switching/routing ASICs, providing wire-speed performance. Packet sampling uses randomness in the sampling process to prevent synchronization with any periodic patterns in the traffic. On average, 1 in every N packets is captured and analyzed. While this type of packet sampling does not provide a 100% accurate result, it does provide a result with quantifiable accuracy. The sFlow packets are encapsulated and sent in UDP over IP (port 6343 by defaut). The sFlow collector receives sFlow data, and generates either a simple-to-parse tagged-ASCII output, or binary output in tcpdump format. To display a real-time trend chart of network traffic, sFlow provides an open source tools called sFlow-rt/flow-trend.

3.2 Congestion monitoring

Detecting congestion in a network is critical for maintaining a stable and reliable network environment. There are different techniques to manage congestion when the congestion is detected such as the use of quality of service (QoS), which prioritize certain types of traffic, or injection throttling. Congestion counters allow us to measure and monitor the level of congestion in a network. Congestion counters in RDMA protocols may vary depending on the implementation and hardware. In order to monitor congestion counters in the switch, we exploit JSON APIs to perform commands remotely from a node in the HPC cluster.

Fig. 1 illustrates our in-network monitoring strategy. We implement our monitoring and visualization in a SmartNIC. Network traffic is observed through the sFlow protocol, and congestion counters are observed through calling JSON API. We visualize the sFlow datagram through sFlow-RT. In order to visualize the congestion counters we implemented a plugin for PcP to support RoCE protocol. So, firstly we implemented new metrics and created PMCD to extract the congestion counters from the switch and visualize them in PcP. Our implementation is publicly available at https://github.com/niks16/iNet.

4 Evaluation

Relocating a monitoring system to network devices poses a significant challenge in terms of allocating resources effectively to optimize the use of resources. To assess the monitoring strategy and its associated overhead, we devised various scenarios for evaluation.

We established a test environment compromising three machines, each equipped with a BlueField-2 operating on Ubuntu 18.04.6. These machines are interconnected by an NVIDIA SN2100 switch through 100Gbps RoCE links, as shown in Fig. 1.

4.1 monitoring

Open Fabrics Enterprise Distribution (OFED) performance test, also called perftest, is a collection of tests intended for performance micro-benchmark of RDMA operations over infiniBand and RoCE protocols [17]. We evaluated the followings tests in our environment between two nodes:

- **ib_write_bw** generates *RDMA WRITE* traffic and calculates the bandwidth. *RDMA WRITE* operation writes to the memory of a remote machine bypassing the CPU and operating system of this remote node.
- **ib_read_bw** generates *RDMA READ* traffic and calculates the bandwidth between a pair of machines. *RDMA READ* reads from the memory of a remote machine, again bypassing the CPU and operating system of the remote node.

We compared the sFlow bandwidth with the OFED performance tests. As can be seen in Table 1, sFlow reports very similar result to perftest.

Table 1: sFlow and perftest comparison.

RDMA operation	sFlow (Gbps)	perftest (Gbps)
ib_write_bw	78.0	79.79
ib_read_bw	80.0	80.85

Furthermore, we conducted testing of our in-network monitoring strategy utilizing the HPC message passing protocol, which leverages RDMA in its underlying layer. Message Passing Interface (MPI) is a popular parallel programming model for scientific applications. The Open MPI is an open-source implementation of the MPI that is developed and maintained by a consortium of academic, research, and industry partners [15]. The Open MPI supports the RDMA UCX library. UCX is an open-source optimized communication library allowing to choose the communication protocols by the runtime characteristics of MPI [21]. Open MPI itself supports different transport protocols to handle communication with different messages sizes [20]. For example, large messages are sent over *Rendezvous protocol*, where the sender first negotiates the buffer availability at the receiver side before the message is actually transferred. In our study, we conducted experiments utilizing the OSU

micro-benchmarks [3], varying the *UCX_RNDV_THRESH* parameter and closely monitoring network traffic. Our observations indicate that UCX adapts its RDMA operations based on the threshold setting. For instance, when setting the threshold to a higher value (e.g., 20M), all messages are transmitted using *RDMA SEND*. *RDMA SEND* represents a two-sided communication approach, necessitating acknowledgment from the receiver before data transmission. This analysis are important as UCX exhibits the ability to dynamically adapt its transport protocol and network interface in response to varying circumstances and not only message size. The utilization of lightweight in-network monitoring facilitates a comprehensive understanding of the underlying MPI communication without imposing any burden on the processes running on the host. Further analysis of *UCX_RNDV_THRESH* can be found on our GitHub repository.

4.2 Monitoring overhead

We evaluated the overhead of our monitoring strategy from augmented network traffic, CPU and memory consumption:

- The network traffic of monitoring depends on the sFlow parameters settings such as sampling rate and sFlow counter-poll-interval. We generated *RDMA READ* traffic through perftest while systematically varying the mentioned parameters and recorded the obtained results. As Fig. 2 (a) and (b) show by increasing the counter-poll-interval and sampling rate the monitoring traffic caused by sFlow decreases.

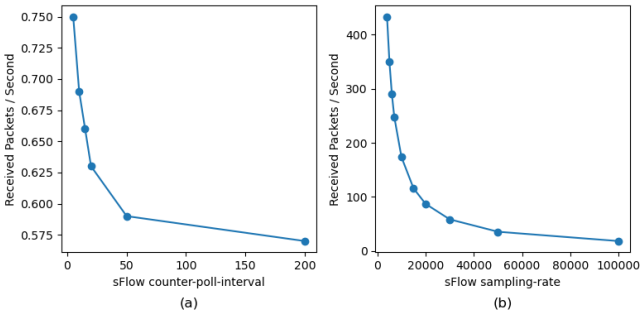

Fig. 2: Network load when using sFlow.

- The processing overhead of the collector on SmartNIC is impressively low, with less than 5% of CPU usage. As for the switch, packet sampling for sFlow is primarily handled by the high-speed switching/routing ASICs, ensuring efficient and swift processing.
- The memory consumed by collector on the SmartNIC was found to be less than 3% on average, which is considerably low.

- When using PcP, the system overhead depends on various factors. These include the number of managed hosts, frequency of logging interval, duration of historical data retention, number of performance metrics measured and application performance [8].

Following the analysis of the monitoring accuracy and its overhead, we have established that the implementation of an accurate lightweight in-network strategy for monitoring RoCE RDMA is feasible and achievable through the utilization of standard protocols. We efficiently utilized only one core of the ARM CPU, leaving a substantial amount of additional resources. Furthermore, the SmartNIC's connection through PCIe to the host enables us to store the observed data at high speeds. This combination of factors enhances our capabilities for in-depth data analysis and processing of observed network traffic.

5 Conclusion and Future work

In this research, different strategies for monitoring RDMA RoCE protocol have been discussed and evaluated to implement an accurate and lightweight in-network monitoring in an HPC cloud setting. In this research, network traffic and congestion from a switch have been monitored. However, as future work we plan to evaluate monitoring of multiple network devices. Moreover, we plan to provide actions based on the network status through some machine learning methods. Additionally, we intend to extend our monitoring scope to include both the network and end nodes using SmartNICs.

Acknowledgment

The work was supported by European Commission under MISO project, (grant 101086541), EEA grants under the HAPADS project (grant NOR/ POLNOR/ HAPADS/ 0049/2019-00) and Research Council of Norway under eX3 project (grant 270053) and Sigma2 (grant NN9342K).

References

1. Abhinav Bhatele, Andrew R Titus, Jayaraman J Thiagarajan, Nikhil Jain, Todd Gamblin, Peer-Timo Bremer, Martin Schulz, and Laxmikant V Kale. Identifying the culprits behind network congestion. In *2015 IEEE International Parallel and Distributed Processing Symposium*, pages 113–122. IEEE, 2015.
2. Alberto Cascajo, Gabriel Gomez-Lopez, Jesus Escudero-Sahuquillo, Pedro Javier Garcia, David E Singh, Francisco Alfaro-Cortés, Francisco J Quiles, and Jesus Carretero. Monitoring infiniband networks to react efficiently to congestion. *IEEE Micro*, 43(2):120–130, 2023.

3. Network-Based Computing Laboratory Dhabaleswar K. (DK) Panda. Mvapich: Mpi over infiniband, omni-path, ethernet/iwarp, roce, and slingshot, 2023.
4. Damu Ding, Marco Savi, Federico Pederzolli, and Domenico Siracusa. Design and development of network monitoring strategies in p4-enabled programmable switches. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6. IEEE, 2022.
5. Network Working Group. https://www.rfc-editor.org/rfc/rfc1157, Accessed: 12-October-2023.
6. Masoud Hemmatpour, Bartolomeo Montrucchio, and Maurizio Rebaudengo. Communicating efficiently on cluster-based remote direct memory access (rdma) over infiniband protocol. *Applied Sciences*, 8(11):2034, 2018.
7. Kyle Hintze, Scott Graham, Stephen Dunlap, and Patrick Sweeney. Infiniband network monitoring: Challenges and possibilities. In *Critical Infrastructure Protection XV: 15th IFIP WG 11.10 International Conference, ICCIP 2021, Virtual Event, March 15–16, 2021, Revised Selected Papers 15*, pages 187–208. Springer, 2022.
8. Edward Jin. Common questions about performance co-pilot. https://www.redhat.com/en/blog/common-questions-about-performance-co-pilot, Accessed: July-2023.
9. Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2014.
10. Neta Ostrovsky Mark Zhang, Erez Alfasi. https://www.man7.org/linux/man-pages/man8/rdma-statistic.8.html, 2019.
11. Priya Mishra, Tushar Agrawal, and Preeti Malakar. Communication-aware job scheduling using slurm. In *Workshop Proceedings of the 49th International Conference on Parallel Processing*, pages 1–10, 2020.
12. NVIDIA. https://docs.nvidia.com/networking/display/bluefielddpuosv380/performance+monitoring+counters#heading-SNMPSubagent, 2023.
13. NVIDIA. https://network.nvidia.com/products/adapter-software/infiniband-management-and-monitoring-tools/, Accessed: July-2023.
14. NVIDIA. Nvidia ethernet switch sdk. https://developer.nvidia.com/networking/ethernet-switch-sdk, Accessed: July-2023.
15. OpenMPI. https://docs.open-mpi.org/en/v5.0.x/index.html, Accessed: July-2023.
16. Performance Co-Pilot (PcP). https://pcp.io/, Accessed: July-2023.
17. Linux RDMA and InfiniBand. Open fabrics enterprise distribution (ofed) performance tests readme. https://github.com/linux-rdma/perftest, Accessed: July-2023.
18. sFlow. Traffic monitoring using sflow. https://sflow.org/sFlowOverview.pdf, 2003.
19. Konstantin S Stefanov, Sucheta Pawar, Ashish Ranjan, Sanjay Wandhekar, and Vladimir V Voevodin. A review of supercomputer performance monitoring systems. *Supercomputing Frontiers and Innovations*, 8(3):62–81, 2021.
20. Sayantan Sur, Hyun-Wook Jin, Lei Chai, and Dhabaleswar K Panda. Rdma read based rendezvous protocol for mpi over infiniband: design alternatives and benefits. In *Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 32–39, 2006.
21. Unified Communication X. https://openucx.org/, Accessed: July-2023.