Network-Centered Resource Management for HPC Networks

Dante Van Poucke, Wouter Tavernier, Didier Colle
IDLab, Ghent University - imec, Ghent, Belgium
{dante.vanpoucke, wouter.tavernier, didier.colle}@ugent.be

***Abstract*—High-Performance Computing (HPC) is indispensable in the current technological era. To preserve the development of innovative ideas and technologies, supercomputers must continue to grow in size. Consequently, the importance of the interconnection network that connects the computing resources, increases. The communication demands of today's workloads can cause bottlenecks in the supercomputer's network, resulting in the utilization of only 3% of their available computing power. This inefficiency presents an opportunity to enhance system efficiency through network-oriented resource management.**

In this paper, we outline the limitations of state-of-the-art resource management strategies and identify the challenges associated with solving the underutilization in large-scale HPC systems. We advocate that flexibility and direct control over resources are the fundamental principles for overcoming these challenges. To this end, we present our research strategy focussing on designing closed-loop resource allocation strategies. Opposed to existing strategies, our approach adaptively reacts to the dynamic behavior of the system through elastic allocation, ensuring optimal resource allocation.

***Index Terms*—HPC, Resource Management, Automatic Optimization, Elastic Jobs.**

I. Introduction

The significance of High-Performance Computing (HPC) cannot be overstated in today's research and industry landscape. Supercomputers play a crucial role in solving the most challenging problems in various domains, such as climate modeling, aircraft design, and decoding the genome of cancerous cells. Further, in the Age of Generative AI, a significant fraction of a supercomputer's compute hours is allocated for deep learning tasks, and this trend is anticipated to continue in an upward trajectory [1].

The power of supercomputers has improved by both vertical and horizontal scaling of the computing components. The horizontal scaling, causes the interconnection network to become one of the most critical components in an HPC system. This is highlighted in Table I, where the theoretical peak performance of the world's largest supercomputers is compared against their actual performance in a well-established HPC benchmark HPCG. The utilization of only 1%-3% is partially due to the communication intensity present in most of today's HPC workloads. For instance, when training Large Language Models, up to 90% of the time is consumed by the communication-intensive AllReduce procedure. The underutilization indicates that an efficient interconnection network is an important factor in fully harvesting the computing power [2]. Moreover, this suggests an opportunity for improved system utilization through better management of network and compute resources.

TABLE I
Comparison between theoretical and actual performance of the world's largest supercomputers

Name	Peak performance (PetaFlop/s)	HPCG performance (PetaFlop/s)	Actual usage
Frontier	1700	14.1	0.83%
Fugaku	537	16	2.98%
LUMI	531	4.6	0.87%
Leonardo	304	3.1	1.02%

Data based on the TOP500 list of November 2023 [3].

Throughout the history of HPC, improvements have been made by relying on dedicated hardware components, but some softwarization techniques have found their way into the HPC environment. The most prominent example is probably Infiniband's subnets manager which enables Software-Defined Networking. Other techniques, like elasticity of resources in cloud computing, are not adopted by the HPC community. We believe that more flexibility and direct control over the system resources will be the key to enabling higher system efficiency. Accordingly, we propose a research strategy to increase HPC system utilization through network-centered resource management. The emphasis on the network is novel with respect to state-of-the-art resource managers which focus solely on compute resources, and is supported by its emergence as the bottleneck in both current and future large-scale supercomputers. Our research aims to enhance the understanding of the impact of resource allocation on network and system performance, ultimately addressing the underutilization of supercomputers. We will develop algorithms for closed-loop resource allocation, enabling a centralized system/network manager to directly optimizes the network. Moreover, we will explore dynamic and elastic allocation strategies, similar to virtual machine allocation, that are capable of adapting their allocation, at runtime, to the dynamic properties of the network and the workloads.

II. Problem Statement

The ultimate objective is to increase the performance of supercomputers by optimizing the network utilization through resource management. This section will describe some important characteristics of HPC networks, formalize the problem of resource management, and identify the key challenges.

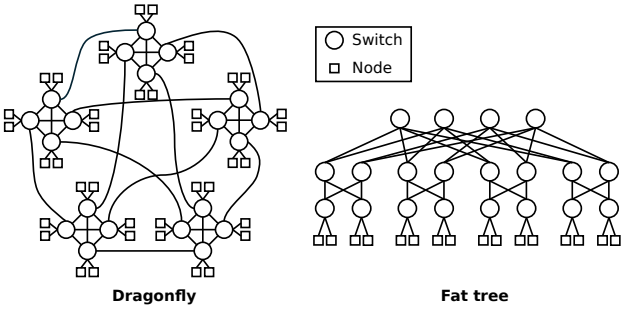

Fig. 1. Example of two typical network topologies for HPC.

A. HPC Networks

The purpose of an HPC interconnection network is to enable collaboration between parallel processes. This network is responsible for scaling the communication to hundreds or thousands of compute nodes while maintaining low-latency and high bandwidth. The most promising network topologies for achieving these requirements are low-diameter topologies, like Dragonfly [4], Jellyfish [5], Slimfly [6] or Polarfly [7]. Compared to fat-tree topologies, which have been the most popular topologies in the past decades, low-diameter topologies lack high path diversity. As a result, static routing strategies are not powerful enough to balance the network traffic, and adaptive routing strategies are preferred in future HPC networks. Current adaptive routing algorithms make decisions based on local information. Every switch will, independent of other switches, select the most suited path for a packet based on the local congestion in the network. Besta et al. foresee an important role for OpenFlow and P4 for further advancements in global adaptive routing for supercomputer networks [8].

However, adaptive routing or other softwarization techniques in networks are a double-edged sword. On the upside, it offers more direct control over the network configuration, leading to improved system performance. However, autonomous decision-making at system level, such as resource management, relies on an accurate network model. Due to the softwarization, the complexity of each layer within the network stack increases significantly, e.g. shifting from static routing to adaptive routing in the network layer. Therefore, constructing such an accurate network model consisting of these layers will become significantly more complex.

Challenge 1: Designing a network model for complex network layers that is both accurate and computationally efficient.

B. Resource Management

Today's supercomputers are no longer dedicated to a single application at any given moment, instead, the system is shared among multiple users. It is the task of the resource manager to monitor and manage the different applications, also called jobs, and resources in the system. For every job, a resource manager needs to make two main decisions: when to start the job (scheduling) and what resources to allocate for the job (resource allocation). The goal is to let multiple jobs make efficient use of the HPC system while maintaining fairness among the users.

A job typically represents a computational task that is divided over multiple processes. These processes are distributed across the computing nodes of the HPC system and typically communicate through MPI. At launch time the user determines the job size, or in other words how many processes will be used. In HPC systems the job size is generally static, and hence, the allocation of the resources is also static. Once resources are allocated they remain fixed for the entire execution of the job. Support for allocation of so-called elastic or malleable jobs, which are jobs that can change their size at runtime, is not available in production HPC systems. The main obstacle to supporting elastic jobs originates in the software stack and especially in the distributed communication libraries, like MPI, which do not contain the flexibility to resize resource pools.

The increased flexibility from considering elastic allocations allows for more dynamic resource allocation and thus better system utilization. Accompanied by this flexibility a new challenge, constraining the search space, emerges. The large scale of today's supercomputers already entails a large search space for static resource management problems. For example, the search space for static resource allocation in Frontier, the most powerful supercomputer in the world containing 9,408 compute nodes, would be $9,408! \approx 10^{76,674}$.

Introducing the freedom of optimizing the job sizes further enlarges this search space. To explore such search spaces, simple meta-heuristics will not suffice, innovative algorithms will be needed to constrain the search space and ensure optimal resource allocation.

Challenge 2: Designing algorithms capable of exploring the large search space of elastic resource allocation in large-scale systems.

C. State-of-the-Art Resource Management

HPC resource management strategies can be grouped into various categories. Approaches relying on **known static properties** reserve resources according to the system's and job's fixed properties, including job size, free cores per node, network topology, and others. The most popular resource manager for supercomputers, the SLURM workload manager, is an example of a resource manager that solely relies on known static properties. Other strategies also include **profiled job characteristics** such as static communication graphs and try to place processes with high affinity close to each other [9], [10], [11]. The last group of approaches is based on **system state information**. LaCurts proposes monitoring throughput between each pair of nodes and allocating heavily communicating processes to node pairs with higher measured throughput

[12]. Zhang proposes assigning congestion-sensitive jobs to compute nodes connected to links with a lower measured traffic intensity [13].

Only a small part of the strategies relies on properties related to the network for making resource management decisions even though the network is often the bottleneck.

Challenge 3: Designing resource managers that rely on network performance for decision-making.

Furthermore, most strategies assume static properties, like fixed job size, static communication pattern, and static routing, while in practice these properties can change during the lifetime of a job.

Challenge 4: Designing resource allocation strategies that can handle the dynamic behavior of the system and the jobs.

III. Research Methodology

Developing our proposed network-centered closed-loop resource allocation strategy requires overcoming the four challenges defined earlier. First, we must identify which network and link attributes are crucial for evaluating the network performance, and rely on these attributes for decision-making (**Challenge 3**). Secondly, we require a network model capable of making efficient and accurate predictions of the previously identified network attributes given a resource allocation (**Challenge 1**). Next, we need an algorithm suitable for making fast resource allocation decisions in large-scale systems (**Challenge 2**). Finally, we must investigate how the closed-loop resource allocator will be able to cope with the dynamic behavior of the system and the jobs (**Challenge 4**). An overview of the closed-loop resource manager is shown in Fig.2.

*A. Identifying crucial network and link attributes (**Challenge 3**)*

A broad spectrum of possible metrics exists to assess the state of a network, encompassing local properties such as link delay, link latency, and link utilization, as well as global properties such as end-to-end delay, tail latency, buffer occupancy, hop count, congestion, and hop-bytes. The values of these metrics are dynamic, can vary over time, and depend on the system's hardware and the active jobs. These metrics are often used in decision-making for various network tasks. In the context of resource allocation, the hob-bytes metric is used to assess process placement due to its strong correlation with the communication time of a job. There is no consensus in literature about a superior performance metric for HPC networks, hence, it will be our task to identify the most suited metrics for resource allocation.

Additionally, we want to be able to measure these metrics at runtime. These measurements will help the resource allocator

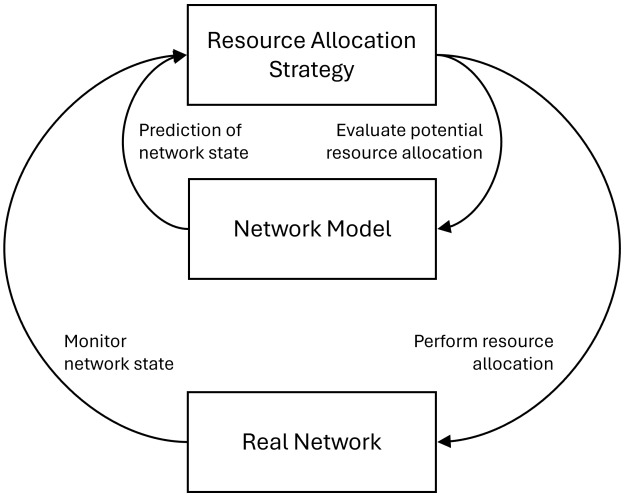

Fig. 2. Closed-loop resource allocator.

to make more informed decisions compared to relying solely on static properties like resource limits, network topology, and communication patterns [13]. Furthermore, leveraging these real-time network measurements offers direct control over the network utilization, and can facilitate a better understanding of the impact of resource allocation on job and system performance. It will be important to investigate the existing monitoring tools embedded in hardware, including switches, routers, and compute nodes, and evaluate their accompanied overhead. Moreover, efficient network protocols will be needed that facilitate the aggregation of local network conditions.

*B. Network model (**Challenge 1**)*

The purpose of the network model is to predict the impact of a job on the system for different sets of allocated resources. It is crucial to be able to predict the changing network conditions to maintain optimal system performance. Predicting these effects is not a straightforward task, as it relies on an accurate model of the entire network stack. For instance, the routing scheme can lead to variations in link utilization despite similar resource allocations. Contrarily, modeling all network details is infeasible, as the evaluation of the network state needs to be swift.

Network modeling is traditionally solved by analytic models (e.g., queueing theory, network calculus) or by using a network simulator. The former approach is often inaccurate due to wrong assumptions or abstractions while the latter approach has high computational cost. With recent innovations in deep learning and Graph Neural Networks (GNN), a new type of network models, like RouteNet [14] and xNet [15], rises which combines the best of both worlds. However, these models have two primary limitations: they can currently only model small networks, and modeling multiple metrics requires the use of multiple separately trained models. Our goal is to develop network models capable of accurately and rapidly predicting essential network metrics for large-scale networks.

C. Making fast and online resource allocation decisions (**Challenge 2**)

The main purpose of a resource manager is to make decisions so that multiple jobs can simultaneously use the HPC system. A resource manager must react as a real-time system with soft deadlines, the usefulness of a decision degrades as soon as another job is finished executing, and the system is underutilized. Resource allocation problems have a search space of $O(n!)$ and are NP-hard. Resorting to approximation algorithms is thus necessary to be able to make fast decisions. Mitigating the scaling problem requires the development of algorithms that can leverage the fact that not all resources are involved in every resource management decisions. To achieve close to optimal results in real-time, these algorithms will need to restrict the search space to a relevant subnetwork and prioritize jobs that are sensitive to network performance.

D. Adapt resource allocation to changing network and link states (**Challenge 4**)

Today's resource allocators are static, they allocate resources at the start of execution and assume that these remain fixed throughout the entire execution. This implies that resource management is inherently greedy. The decisions that are optimal at launch time may become suboptimal as the system's status changes due to the introduction or removal of jobs, hardware failures, a scenario that is encountered frequently in large-scale systems, or incorrect initial predictions about the job and system characteristics. Escaping these suboptimal configurations requires the reallocation of a job's resources, involving the migration of code and data to different resources, thereby introducing additional overhead.

The concept of malleable jobs, jobs for which the number of used resources can be dynamically adjusted at runtime, represents a first strategy to adapt to changing system conditions. Similarly to Virtual Machine (VM) migration in data centers, the migration of HPC jobs introduces new scheduling possibilities that can enhance the efficiency of the supercomputers [16]. As support for managing dynamic jobs in HPC environments is relatively limited, the scheduling opportunities have not been fully explored.

An important aspect of elastic job placement involves assessing the overhead accompanied by adapting the resource allocation at runtime. Additionally, the benefits of adapting the resource allocation at runtime can be evaluated by leveraging the network model. Following the evaluation of the overhead and gains associated with dynamic resource allocation, we can use the resource allocator, described in the previous subsection, to perform resource reallocation. However, this will require several modifications, first, due to the considerably larger search space, and second, because of the need to determine optimal timing for exploring improved reallocations. Our objective is to further explore the potential of elastic resource allocation and map the overhead and gains involved in scheduling at runtime. We aim to do this in a network-informed manner, by using live monitored data of the crucial network metrics.

IV. Conclusion

This paper presented a research strategy for a network-centered closed-loop resource allocator in HPC systems. We argued that more direct control of the network configuration will enable higher network utilization. Similarly, switching from static to elastic resource allocation enhances the capability to adapt to changing system conditions, leading to more efficient usage of supercomputers.

References

[1] Hyperion Research, "HPC Market Update: HPC/AI Market Results, and High Growth Areas," https://hyperionresearch.com/hpc-market-update-briefing-during-sc23-virtual/, Nov. 2023.

[2] D. De Sensi, S. Di Girolamo, K. H. McMahon, D. Roweth, and T. Hoefler, "An In-Depth Analysis of the Slingshot Interconnect," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2020, pp. 1–14.

[3] "Top500: Top 500 supercomputers, November 2023," https://www.top500.org/.

[4] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-Driven, Highly-Scalable Dragonfly Topology," in *2008 International Symposium on Computer Architecture*. Beijing, China: IEEE, Jun. 2008, pp. 77–88.

[5] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking Data Centers Randomly."

[6] M. Besta and T. Hoefler, "Slim Fly: A Cost Effective Low-Diameter Network Topology," Jun. 2020.

[7] K. Lakhotia, M. Besta, L. Monroe, K. Isham, P. Iff, T. Hoefler, and F. Petrini, "PolarFly: A Cost-Effective and Flexible Low-Diameter Topology," in *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2022, pp. 1–15.

[8] M. Besta, J. Domke, M. Schneider, M. Konieczny, S. D. Girolamo, T. Schneider, A. Singla, and T. Hoefler, "High-Performance Routing With Multipathing and Path Diversity in Ethernet and HPC Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 4, pp. 943–959, Apr. 2021.

[9] T. Hoefler and M. Snir, "Generic topology mapping strategies for large-scale parallel architectures," in *Proceedings of the International Conference on Supercomputing*, ser. ICS '11. New York, NY, USA: Association for Computing Machinery, May 2011, pp. 75–84.

[10] Y. Baicheng, Y. Zhang, X. Limin, Z. Yi, W. Bing, and S. Yao, "LPMS: A Low-cost Topology-aware Process Mapping Method for Large-scale Parallel Applications on Shared HPC Systems," in *Workshop Proceedings of the 48th International Conference on Parallel Processing*, ser. ICPP Workshops '19. New York, NY, USA: Association for Computing Machinery, Aug. 2019, pp. 1–10.

[11] E. Jeannot, "Process mapping on any topology with TopoMatch," *Journal of Parallel and Distributed Computing*, vol. 170, p. 39, 2022.

[12] K. LaCurts, S. Deng, A. Goyal, and H. Balakrishnan, "Choreo: Network-aware task placement for cloud applications," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: Association for Computing Machinery, Oct. 2013, pp. 191–204.

[13] Y. Zhang, B. Aksar, O. Aaziz, B. Schwaller, J. Brandt, V. Leung, M. Egele, and A. K. Coskun, "Using Monitoring Data to Improve HPC Performance via Network-Data-Driven Allocation," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, Sep. 2021, pp. 1–7.

[14] M. Ferriol-Galmés, J. Paillisse, J. Suárez-Varela, K. Rusek, S. Xiao, X. Shi, X. Cheng, P. Barlet-Ros, and A. Cabellos-Aparicio, "RouteNet-Fermi: Network Modeling with Graph Neural Networks," *IEEE/ACM Transactions on Networking*, vol. 31, no. 6, pp. 3080–3095, Dec. 2023.

[15] M. Wang, L. Hui, Y. Cui, R. Liang, and Z. Liu, "xNet: Improving Expressiveness and Granularity for Network Modeling with Graph Neural Networks," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, May 2022, pp. 2028–2037.

[16] M. Rodríguez-Pascual, J. Cao, J. A. Moríñigo, G. Cooperman, and R. Mayo-García, "Job migration in HPC clusters by means of checkpoint/restart," *The Journal of Supercomputing*, vol. 75, no. 10, pp. 6517–6541, Oct. 2019.