Bandwidth Allocation in Silicon-Photonic Networks Using Application Instrumentation

Aditya Narayan, Ajay Joshi and Ayse K. Coskun
Boston University, Boston, MA 02215, USA; E-mail: {adityan, joshi, acoskun}@bu.edu

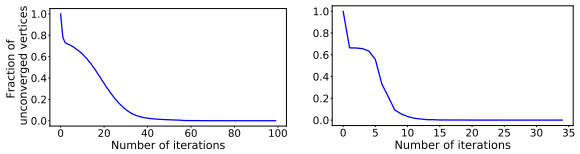
***Abstract*—Photonic network-on-chips, despite their low-latency and high-bandwidth-density advantages in large manycore systems, suffer from high power overhead. This overhead is further exacerbated by the high bandwidth demands of data-centric applications. Prior works utilize bandwidth allocation policies at system-level to minimize photonic power and provide required bandwidth for applications. We present an approach to minimize the bandwidth requirements by instrumenting an application at the software level. This instrumented information is used to assist bandwidth allocation at system-level, thereby reducing the photonic power. We instrument PageRank application and demonstrate 35% lower power using instrumentation-assisted bandwidth allocation on PageRank running real-world graphs compared to bandwidth allocation on uninstrumented PageRank.**

I. Introduction

Emerging data-centric applications such as graph algorithms have large memory footprints and exhibit high parallelism resulting in high on-chip communication traffic. Silicon-photonics has paved the way for low-latency and high-bandwidth-density communication links in large manycore chips [1], [2]. However, the photonic power resulting from laser and thermal tuning grows with the provided on-chip bandwidth and more than offsets the latency and bandwidth advantages [3].

Several approaches at system-level utilize bandwidth allocation policies to address the bandwidth-power tradeoff in photonic networks (e.g. [4]–[6]). Bandwidth allocation is typically performed by assigning optical wavelengths depending on the bandwidth requirements. These bandwidth allocation policies in photonic networks include offline characterization to determine balanced bandwidth [4], analysis of application task graphs [5], or hardware reconfiguration to support the required bandwidth [6]. The on-chip communication traffic, however, highly depends on the software implementation of the application algorithm. This dependence provides an opportunity to develop a generalized software-level approach for bandwidth allocation in photonic networks. None of the prior works have explored the use of application instrumentation to perform bandwidth allocation.

We propose to instrument an application at the software-level to guide bandwidth allocation at the system-level. We instrument data structures or privileged instructions in the application source code to provide information regarding the communication traffic and bandwidth density required for an application. This information can then be utilized at the system level to perform bandwidth allocation.



(a) (b)

Fig. 1. Number of unconverged vertices with iterations for PageRank on (a) Google webgraph, (b) Kronecker graph with 2^{18} vertices.

In this work, as an example, we consider PageRank, an iterative graph algorithm that has extremely high parallelism. We demonstrate that using appropriate instrumentation of PageRank source code, we pass additional information regarding active vertices that can be used to reduce the network bandwidth density. We leverage this information to perform bandwidth allocation as proposed by Narayan *et al.* [4] (*WAVES*) and demonstrate 35% higher power savings compared to bandwidth allocation on uninstrumented PageRank.

II. Motivational Example: PageRank

PageRank is used on web graphs to determine the quality of a vertex. It assigns a rank to the vertex based on the number of connected vertices and their associated ranks. It is an iterative algorithm to converge the rank values of vertices. The algorithm begins with equal ranks assigned to each vertex in the input graph. Depending on the number of vertices connected to a vertex v, ($g.out\_degree(v)$), the rank of v is updated. At the end of every iteration, the rank of each vertex is compared with an error threshold. The algorithm iterates until all vertices converge.

A key characteristic of PageRank is the varying number of iterations required to converge the vertices, which result in asymmetric convergence [7]. We demonstrate this characteristic by running PageRank on a Google webgraph from SNAP [8] and a Kronecker graph with 2^{18} vertices. Figure 1 shows the fraction of vertices that have not yet converged at the end of each iteration. On a Google webgraph, 21.77% of vertices converge in a single iteration, another 75.8% of vertices converge in the next 40 iterations, and less than 3% of vertices converge in the last 60 iterations. It can be noted that a significantly high fraction of vertices converge in the first few iterations, leaving a very low fraction of unconverged vertices in later iterations. This observation implies reduced memory accesses in later iterations. Thus, there is an opportunity to reduce the network bandwidth between memory and LLCs by deactivating certain photonic links in later iterations and save photonic network power.

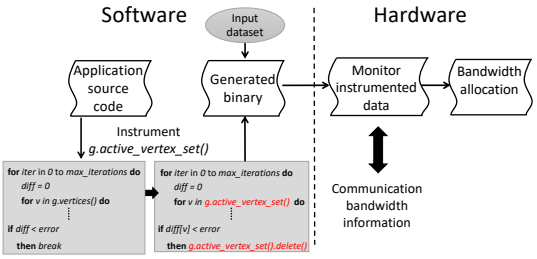
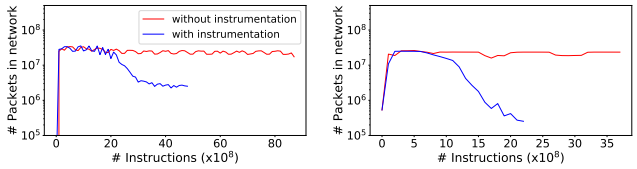


Fig. 2. Application instrumentation-assisted bandwidth allocation.



(a) *Google webgraph* (b) *Kronecker graph (2^{18} vertices)*

Fig. 3. Number of packets transferred in the photonic network during application execution.

III. Application Instrumentation

Figure 2 shows the framework of our proposed application instrumentation-assisted bandwidth allocation. As PageRank is characterized by asymmetric convergence of vertices, we maintain a data structure called active vertex set [7]. The active vertex set maintains a list of all unconverged vertices. During each iteration, PageRank algorithm operates only on vertices in the active vertex set. At the end of each iteration, we update this active vertex set by deleting vertices that converge during the current iteration. Figure 1 shows that the size of the active vertex set decreases with increasing iteration count.

We study the network characteristics when an instrumented PageRank is executed on a 2.5D system with photonic networks [9]. Figure 3 illustrates the network packets transferred in the photonic network during application execution. In uninstrumented PageRank, the algorithm operates on all the vertices during every iteration, causing the network packets transferred in the photonic link to remain roughly constant. In contrast, since we remove the converged vertices from the active vertex set in instrumented PageRank, the algorithm operates on a lower fraction of vertices every iteration. This results in an overall decrease in the LLC and main memory communication, resulting in lower number of packets in the photonic network as the application progresses. At the system-level, we monitor the number of active vertices in PageRank as application progresses. The bandwidth allocation policy uses this instrumented information in addition to other network parameters to allocate the required bandwidth.

IV. Evaluation

We evaluate a prior bandwidth allocation policy, WAVES [4], on instrumented and uninstrumented PageRank algorithm. We model a 2.5D manycore system with photonic networks [9] in Sniper [10]. We evaluate the performance of PageRank algorithm from GAP-BS benchmark [11] on Kronecker graphs and real-world graphs [8]. We modify the source code in PageRank to maintain the active vertex set that is updated every iteration with unconverged vertices.

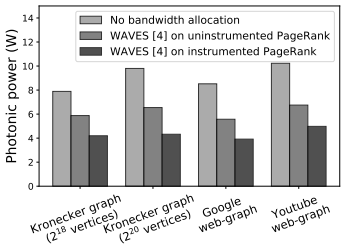


Fig. 4. Photonic power savings using application instrumentation-assisted bandwidth allocation

We model different network bandwidths in Sniper for instrumented and uninstrumented PageRank and determine the network bandwidth that satisfies the performance loss threshold as modeled in WAVES [4]. We calculate the photonic network power using an analytical model [4]. Figure 4 shows the photonic power savings compared to a system with no bandwidth allocation.

For instrumented PageRank, WAVES allocates a higher bandwidth for initial iterations as more vertices converge, and the bandwidth is reduced for later iterations. As a result, the photonic power, which is proportional to the provided bandwidth, is reduced. Using our instrumentation-assisted bandwidth allocation across four datasets, on average, we reduce 35.13% of photonic network power compared to bandwidth allocation on an uninstrumented PageRank algorithm.

V. Conclusion

While system-level bandwidth allocation policies have addressed power-bandwidth tradeoff in photonic networks, we demonstrate that software-level instrumentations can further assist these policies to reduce photonic network power. As an example, we demonstrate the benefits of instrumented-assisted bandwidth allocation on PageRank. Our approach is scalable and can be generalized to other high-bandwidth applications running on manycore systems with photonic networks.

Acknowledgement

This work was funded by NSF CCF-1716352.

References

[1] Y. Demir *et al.*, "Galaxy: A high-performance energy-efficient multi-chip architecture using photonic interconnects," in *Proc. ICS*, 2014.
[2] A. Narayan *et al.*, "System-level evaluation of chip-scale silicon photonic networks for emerging data-intensive applications," in *Proc. DATE*, 2020.
[3] M. Bahadori *et al.*, "Energy-bandwidth design exploration of silicon photonic interconnects in 65nm CMOS," in *Proc. OI*, 2016.
[4] A. Narayan *et al.*, "WAVES: Wavelength selection for power-efficient 2.5D-integrated photonic NoCs," in *Proc. DATE*, 2019.
[5] J. Luo *et al.*, "Performance and energy aware wavelength allocation on ring-based WDM 3D optical NoC," in *Proc. DATE*, 2017.
[6] C. Chen *et al.*, "Managing laser power in silicon-photonic NoC through cache and NoC reconfiguration," *IEEE TCAD*, 2015.
[7] M. M. Ozdal *et al.*, "Architectural requirements for energy efficient execution of graph analytics applications," in *Proc. ICCAD*, 2015.
[8] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.
[9] Y. Thonnart *et al.*, "POPSTAR: A robust modular optical NoC architecture for chiplet-based 3D integrated systems," in *Proc. DATE*, 2020.
[10] T. E. Carlson *et al.*, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *Proc. SC*, 2011.
[11] S. Beamer, K. Asanović, and D. Patterson, "The gap benchmark suite," *arXiv preprint arXiv:1508.03619*, 2015.