Effective Cache Bank Placement for GPUs

Mohammad Sadrosadati*, Amirhossein Mirhosseini†, Shahin Roozkhosh*, Hazhir Bakhishi* and Hamid Sarbazi-Azad*‡
*Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
†Department of Electrical Engineering & Computer Science, University of Michigan, Ann Arbor, USA
‡Computer Science School, Institute for Researches in Fundamental Sciences, Tehran, Iran
Emails: {sadrosadati,roozkhosh,bakhishi}@ce.sharif.edu, miramir@umich.edu, azad@{sharif.edu, ipm.ir}

***Abstract*—The placement of the Last Level Cache (LLC) banks in the GPU on-chip network can significantly affect the performance of memory-intensive workloads. In this paper, we attempt to offer a placement methodology for the LLC banks to maximize the performance of the on-chip network connecting the LLC banks to the streaming multiprocessors in GPUs. We argue that an efficient placement needs to be derived based on a novel metric that considers the latency hiding capability of the GPUs through thread level parallelism. To this end, we propose a throughput aware metric, called Effective Latency Impact (ELI). Moreover, we define an optimization problem to formulate our placement approach based on the ELI metric mathematically. To solve this optimization problem, we deploy a heuristic solution as this optimization problem is NP-hard. Experimental results show that our placement approach improves the performance by up to 15.7% compared to the state-of-the-art placement.**

I. Introduction

Off-chip memory access latency is one of the known sources of performance degradation in computer systems. In GPUs, this problem is mitigated through Thread Level Parallelism (TLP) in a way that when a thread (or a group of threads known as a warp) suffers from a long latency memory access, it is preempted by the scheduler and another thread starts executing until the data needed by the first one is ready. However, research studies [8] show that due to the limitations of various resources such as on-chip storage units and memory bandwidth, off-chip memory latency cannot be thoroughly concealed via TLP.

Last Level Cache (LLC) is one of the main resources in the GPU memory systems that plays a critical role in the memory access latency. LLC banks are connected to Streaming Multiprocessor (SM) units through an on-chip network that should be able to provide large amounts of bandwidth compared to a conventional multiprocessor interconnect [3]. The traffic over GPU on-chip networks (GoNs) has a many-to-few pattern [6], [3]. This is because, first, there are two communication flow types: SM to LLC and LLC to SM [6]; and second, the number of LLC banks is much smaller than the number of SMs as can be seen in Figure 1. As a result, the blocking rate is very high around the LLC banks and they have a high probability of becoming hot-spots in the network as shown in Figure 2.

The placement of the LLC banks across the on-chip network can considerably influence the traffic distribution and hence, the average LLC access latency [1], [6]. To illustrate the effect of the LLC bank placement on the traffic distribution, we have conducted cycle-accurate simulations for two different LLC placements known as *Bottom* and *Top-Bottom*, and calculated the traffic distribution for each. As shown in Figure 2, the *Bottom* placement results in the accumulation of the traffic on one side of the network while the *Top-Bottom* placement attempts to distribute the traffic loads over the whole network. Our experimental results show that up to 45% IPC movement can be achieved by *Top-Bottom* compared to *Bottom* placement.

Previous studies have considered the average hop count in the network as a useful metric for devising an efficient LLC bank placement in GPUs. Nonetheless, we argue that, unlike in conventional Chip-Multiprocessors (CMPs), average hop count cannot be beneficial when finding efficient LLC placements in GPU architectures for two reasons: First, due to the large traffic load and potential hot-spots of GoNs, number of hops does not give any information about the packet latencies in such networks. In other words, it is very likely that packets taking longer routes have smaller latencies as they travel through less congested areas. Second, even larger average memory access times do not necessarily translate to lower performance in GPUs as they are designed to hide the memory latencies through TLP. As a result, classic metrics such as average hop count proposed by previous studies cannot be useful for evaluating different LLC bank placements in GoNs.

In this paper, we introduce a novel metric, called Effective Latency Impact (ELI), for evaluating different LLC bank placements in GoNs. We use this metric to propose a systematic methodology for finding efficient placements with different numbers of SMs and LLC banks. To this end, we define a synthetic traffic pattern that models GPU memory-intensive workloads and an optimization problem that we solve using a heuristic search algorithm. Finally, we evaluate our methodology by comparing the resulted placements with the LLC bank placements that exist in the state-of-the-art architectures. Our experimental results show that this method outperforms the existing placements by up to 15.7% in terms of performance under various workloads.

II. Background and Motivation

The first level of the GPU memory system hierarchy is a fast and small size L1 cache which is embeded inside the SMs. The access time of the L1 cache is close to the one of the register file. The L1 cache can also be configured to be

978-3-9815370-8-6/17/$31.00

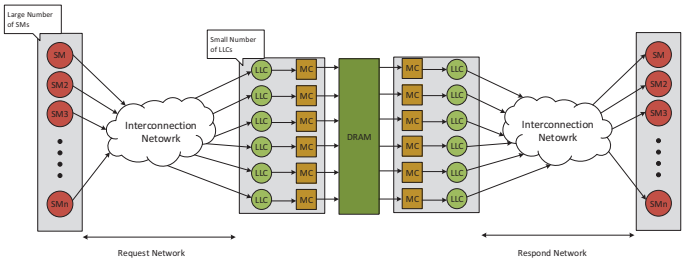

Fig. 1. A high-level view of GoN. Many SMs send requests to few LLC banks and few LLC banks respond to many SMs.

(a) Bottom (b) Top-Bottom

Fig. 2. The traffic load distribution for two different placements of the LLC bank; (a) *Bottom*, (b) *Top-Bottom*

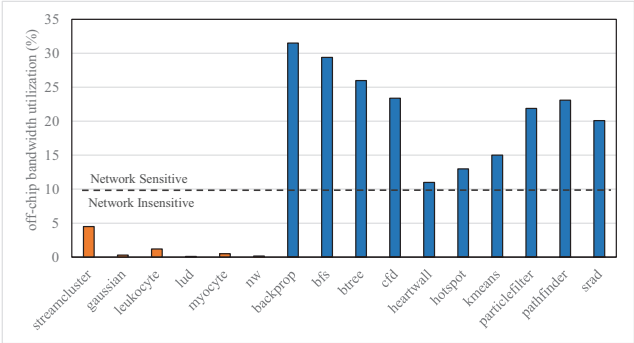

Fig. 3. Off-chip bandwidth utilization for Rodinia [5] workloads (LLC is disabled).

used as a scratchpad memory shared between all of the threads that are scheduled on that SM. Moving farther from the L1 cache, there are some LLC banks that are shared between all of the SMs and are connected to the SMs through an on-chip network. LLC banks are coupled to memory contollers in a one to one correspondence manner. In other words, each LLC bank is only responsible for caching the part of the address space that corresponds to its own memory controller. Each LLC bank grouped with its correspondent memory controller forms a memory node.

Only the memory-intensive workloads with low L1 hit rates are able to magnify the contribution of GoN on the total GPU performance as most of their memory accesses are serviced either by LLC or the off-chip memory. We call such workloads network-sensitive applications. In order to classify different applications into network-sensitive and network-insensitive categories, we conducted simulations for various workloads and calculated the off-chip memory bandwidth utilization for them. Since a large portion of the memory accesses are filtered by the LLC, we have disabled the LLC in our experiments. As depicted in Figure 3, almost 63% of the workloads from Rodinia benchmark suite [5] are network-sensitive. As a result,

TABLE I

THE TRAFFIC RATE FOR THE DIFFERENT FLOWS IN OUR SYNTHETIC TRAFFIC MODELING MEMORY-INTENSIVE GPU WORKLOADS. ($k = \beta\gamma$, $\beta = \#SM/\#LLC$, $\gamma = 0.35$)

flow type	rate
SM-to-SM	0
SM-to-LLC	λ
LLC-to-SM	$k\lambda$
LLC-to-LLC	0

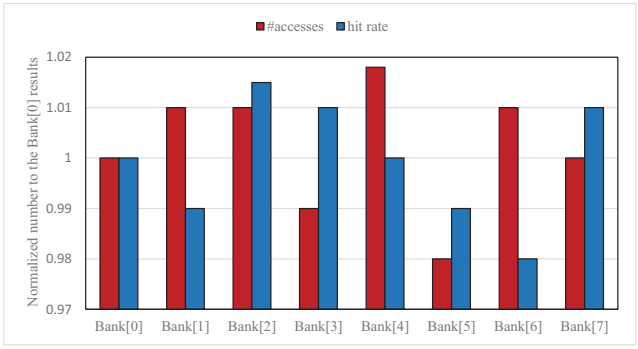

Fig. 4. number of the accesses and the hit rates are almost identical among different LLC banks.

such workloads should be given enough attention as the total performance of the GPU is significantly influenced by the GoN when the application is network-sensitive. In the rest of the paper, we have selected some random network sensitive workloads from ISPASS [4], Rodinia [5], and Parboil [15] benchmarks and performed our experiments on them.

An efficient LLC bank placement plays a crucial role in reducing the GoN latency for network-sensitive workloads. The state-of-the-art solutions proposed for bank placement in GoNs attempt to minimize the average hop count to improve the network performance [6]. However, the effect of the average number of hops on the performance of the on-chip network is non-trivial in high traffic loads with large blocking rates. For near-saturation rates, load balancing is more effective for reducing the average packet latency rather than minimizing the average hop count [1]. Moreover, network latency cannot individually be used as a raw metric for optimizing the LLC placement in throughput workloads and architectures as they are designed to be able to hide the memory latency using TLP. Therefore, a more efficient placement methodology of LLC banks is required to balance the link utilizations and reduce the network blocking rate.

In this paper, we propose a novel placement methodology for the LLC banks based on a new metric which specifically targets network-sensitive workloads and attempts to improve their performance by reducing the network blocking rate.

III. PROPOSED METHOD

In this section, we propose a novel methodology for the placement of the LLC banks in GoNs. To this end, we define a synthetic traffic pattern that models the behavior of the network-sensitive GPU workloads and use it to formulate an

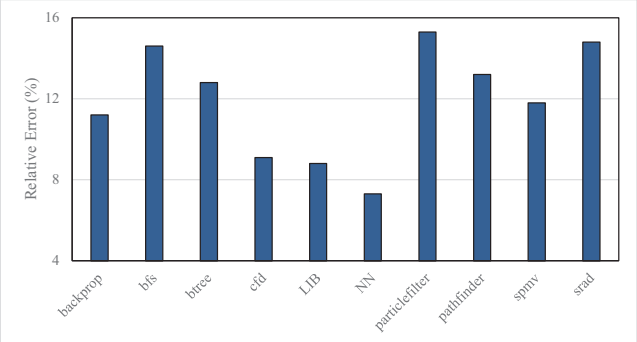

Fig. 5. The relative error for the proposed synthetic traffic pattern which models the on-chip traffic behavior.

optimization problem. Finally, we generate an efficient placement by solving the optimization problem using a heuristic search algorithm.

A. GPU synthetic traffic pattern

We define a synthetic traffic pattern which models the GoN behaviour of the network-sensitive throughput workloads. This synthetic traffic pattern gives us the opportunity to analyze different placement approaches with low cost. We assume that the average injection rate of the LLC banks is larger than the average injection rate of the SMs by the ratio of *K* which is composed of two components β and γ. β denotes the ratio of the number of SMs to the number of LLCs and γ indicates different application characteristics that affect *K* (e.g. ratio of the number of reads to the number of writes). In other words, *K* is composed of two components β and γ where β models the network and γ models the workload characteristics.

Moreover, our experimental results in Figure 4 reveal that both the number of the accesses and hit rate are almost identical among different LLC banks. As a result, in our synthetic traffic pattern, we select the destination of read-request and write-request packets from LLC banks and the destinations of read-reply and write-reply packets from SMs randomly with uniform distribution. Table I reports the traffic rate of the different flows in the GoN. As explained before, the injection rate of the flows originated from LLCs is *K* times larger than the injection rate of the ones originated from SMs. The β parameter has to be calculated and set based on the ratio of the number of SMs to the number of LLCs in the network topology. The γ parameter, on the other hand, should be the average of γ parameters extracted from all workloads we are considering as reference points. The way we have extracted the γ parameter for each workload is by conducting an initial full system simulation and extracting the *K* parameter by dividing the average injection rate of LLCs by the average injection rate of SMs. The γ parameter can then be calculated by dividing *K* by β. The γ parameter we have extracted based on our targeted network-sensitive workloads is 0.35.

To justify the proposed synthetic traffic pattern, we calculated the injection rates of SMs, the average packet latencies, and the network throughput for network-sensitive GPU workloads using the network simulator embedded in GPGPU-Sim simulation environment. GPGPU-Sim uses Booksim 2.0 [7] as a well-known network simulator with few modifications. We implemented our proposed synthetic traffic pattern which models the traffic behavior of network-sensitive GPU workloads in Booksim 2.0 and measured the average packet latency and throughput for different SM injection rates. Finally, we compared our results for the average packet latency and network throughput with their accurate values extracted using GPGPU-Sim and calculated the relative errors. Results (the relative errors) are reported in Figure 5. As shown in this figure, the relative error is at most 15.3% which is well acceptable for a synthetic traffic pattern modeling real applications (note that we have considered γ = 0.35 for all workloads).

B. Problem formulation

As discussed before, a new placement methodology for the GPU LLC banks is essential in order to reduce the network blocking rate. The most accurate way to find the best placement of LLC banks is to conduct simulations for all placement approaches and calculate IPC for each placement in order to find the placement that can result in the highest performance. However, orchestrating cycle accurate simulations for each placement in order to calculate the IPC is very time-consuming and thus impractical. Therefore, we formulate an optimization problem to determine the optimum placement for LLC banks using a novel metric that particularly attempts to improve the performance of the network-sensitive throughput workloads. This metric, called ELI, estimates the weighted average packet latency of the network with larger weights for the higher latencies as throughput architectures, such as GPUs, are designed to be able to hide small latencies through TLP. The problem explained here is inspired from [14].

We assumed that the GoN is a 2D mesh. Given that the network topology is a graph G(V,E) where V is the set of the vertices and E is the set of the ordered pairs (u,v) indicating the directed network links. Assume that F is the set of different f_i values where f_i is a communication flow among the i-th source-destination pair (src_i, dst_i) with the traffic rate r_i. Moreover, we employed a boolean variable b_i(u, v) which denotes if the flow f_i passes through the directed link (u,v). The problem should guarantee that it avoids the violation of the following constraints:

$$\forall f_i \in F \sum_{(src_i,v)\in E} b_i(src_i, v) = 1 \tag{1}$$

$$\forall f_i \in F \sum_{(u,dst_i)\in E} b_i(u, dst_i) = 1 \tag{2}$$

$$\forall f_i \in F \wedge v \notin \{src_i, dst_i\} \quad \sum_{(u,v)\in E} b_i(u, v) = \sum_{(v,w)\in E} b_i(v, w) \tag{3}$$

$$\forall f_i \in F \sum_{(u,v)\in E} b_i(u, v) = distance_i \tag{4}$$

Equation 1 denotes that f_i goes through only one of the output links of its source node. Equation 2 denotes that f_i is received from only one of the input links of its destination node. Equation 3 denotes that when f_i enters an intermediate node (neither its source nor its destination), it will finally get out of that node. Equation 4 denotes that f_i has to only travel $distance_i$ which is the minimum distance between src_i and dst_i.

Since XY routing function is deployed to route the flow f_i, North-East (NE), North-West (NW), South-East (SE), and South-West (SW) turns are illegal. Equations 5, 6, 7, and 8 formulate the prohibition of NE, NW, SE, and SW turns respectively. Furthermore, flows are not allowed to have any U-turns. This is formulated in Equation 9.

$$\forall f_i \in F \wedge \forall x \in [0, n-2] \wedge \forall y \in [1, n-1] \quad b_i((x, y-1), (x, y)) + b_i((x, y), (x+1, y)) \leq 1 \tag{5}$$

$$\forall f_i \in F \wedge \forall x \in [1, n-1] \wedge \forall y \in [1, n-1] \quad b_i((x, y-1), (x, y)) + b_i((x, y), (x-1, y)) \leq 1 \tag{6}$$

$$\forall f_i \in F \wedge \forall x \in [0, n-2] \wedge \forall y \in [0, n-2] \quad b_i((x, y+1), (x, y)) + b_i((x, y), (x+1, y)) \leq 1 \tag{7}$$

$$\forall f_i \in F \wedge \forall x \in [1, n-1] \wedge \forall y \in [0, n-2] \quad b_i((x, y+1), (x, y)) + b_i((x, y), (x-1, y)) \leq 1 \tag{8}$$

$$\forall f_i \in F \wedge \forall x \in [1, n-1] \wedge \forall y \in [0, n-1] \quad b_i((x-1, y), (x, y)) + b_i((x, y), (x-1, y)) \leq 1 \tag{9}$$

As explained before, we aim to minimize the ELI metric that gives larger weights to higher packet latencies. To this end, we have to derive an analytical representation of the packet latency. This is formulated in Equation 11 which uses the output of Equation 10. Equation 10 calculates the loads of different links as a representation of their latency. Finally, Equation 12 attempts to calculate the ELI metric which is the cost function. The α parameter in this equation determines the degree by which we give higher weights to larger latencies. Our experiments (details in Section IV) show that this parameter should be set to larger values when the network has higher SM-to-LLC ratios.

$$\forall u, v \in E \quad L(u, v) = \sum_i r_i \times b_i(u, v) \tag{10}$$

$$\forall f_i \in F \quad T_i = \sum_{(u,v) \in E} b_i(u, v) \times L(u, v) \tag{11}$$

$$\forall f_i \in F \quad ELI = \frac{\sum_i T_i^{\alpha}}{\#flows} \tag{12}$$

TABLE II
THE MAIN SIMULATION PARAMETERS IN GPGPU-SIM.

Number of SMs	24, 32, or 56
Core clock	700 MHz
Scheduler	Two-level
Number of warps per SM	48
L1D Cache	4-way, 16 KB, 128 B line
L1I Cache	2 KB, 128 B line
LLC	8-way, 1 MB, 128 B line
Number of LLC banks	8
Number of memory controllers	8
Network topology	2D Mesh
Flit size (bits)	128
Routing function	XY

C. Heuristic solution

Since the discussed optimization problem in the previous section is an NP-hard problem, we deployed a Simulated Annealing Algorithm (SA) to offer a fast heuristic solution for the placement of the LLC banks. SA is a heuristic search technique in computer science to find a best-effort solution for an optimization problem and is widely used for various types of placement problems with different cost functions. SA attempts to minimize the value of the cost function by mimicking the cooling process of a multiatomic system. The slow cooling process is interpreted as a slow decrease in the probability of accepting worse solutions while exploring the solution space (continuously replacing two random blocks and evaluating the new placement). Accepting worse solutions is necessary as it allows for a more extensive search in order to find a globally optimal solution.

IV. EVALUATION

A. Simulation environment

We evaluated the performance of the proposed placement approach using GPGPU-Sim [4]. The main configuration parameters of GPGPU-Sim are provided in Table II. We have chosen the baseline placement to be the *Top-Bottom* placement and normalized our results based on it. Moreover, Since the previous works have considered the average hop count as an important metric for selecting an efficient placement [6], we have also run our SA with a modification in the cost function to find a placement with the minimum average hop count. This placement, called *MinHop*, is generated with SA using a cost function that evaluates different placements by the average hop count. This cost function is calculated as:

$$H_{avg} = \frac{\sum(H_{vertical} + H_{horizontal})}{\#flows} \tag{13}$$

where, H_{avg} shows the average hop count, $H_{vertical}$ and $H_{horizontal}$ denote vertical and horizontal hop counts for each flow, respectively. We have compared our approach, called *MinELI*, with the *MinHop* placement in order to show that the average hop count is not a reliable metric for throughput architectures and workloads. Note that, the average hop count of *MinHop* is 8% less than the average hop count

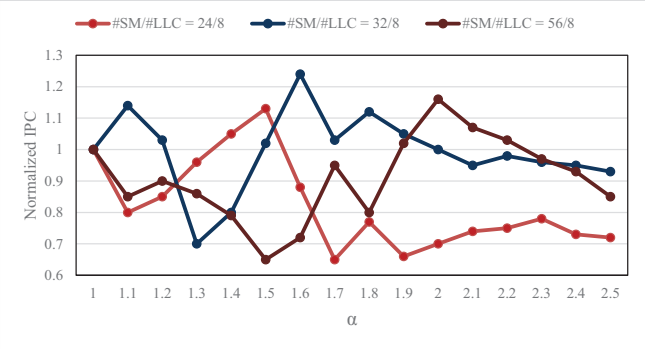

Fig. 6. Normalized IPC results using different SM/LLC ratios for various values of the α parameter.

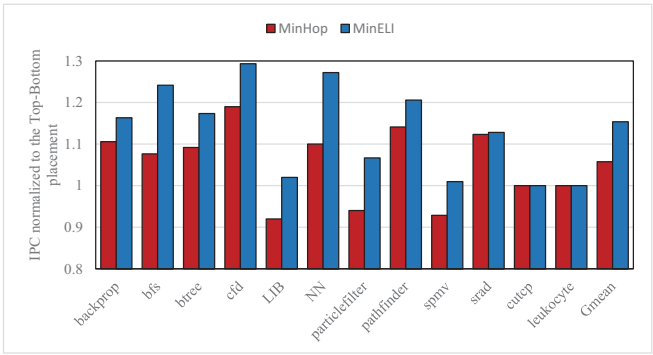

Fig. 7. Normalized IPC results of *MinHop* and *MinELI* with respect to *Top-Bottom* placement (#SM/#LLC = 24/8).

of the *MinELI* placement. We calculated the IPC as our comparison metric. Furthermore, we used several network-sensitive GPGPU workloads from ISPASS [4], Rodinia [5], and Parboil [15] benchmark suits as well as two network-insensitive workloads in order to analyze the performance of the proposed method under different classes of applications.

B. Tuning the α parameter

As explained before, the ELI metric gives larger weights to higher packet latencies in the network. This degree is tunable by the α parameter in Equation 12. Our observations show that the α parameter should be set to higher values when the ratio of the number of SMs to the number of LLCs is larger. The experimental results in Figure 6 report the highest achievable IPC with different ratios of the number of SMs to the number of LLCs and different values of α. We have used the best α values extracted by experiments for each ratio.

C. Performance analysis

We measured IPC values using GPGPU-Sim for *MinELI* placement and compared it against *MinHop* and *Top-Bottom* placements under several workloads. Results are depicted in Figure 7. Note that in this figure, numbers are normalized to *Top-Bottom* placement results. As shown in the figure, our method improves the IPC by up to 29.3% and 15.7% (15.3% and 9.1%, on average) compared to *Top-Bottom* and *MinHop* placements, respectively.

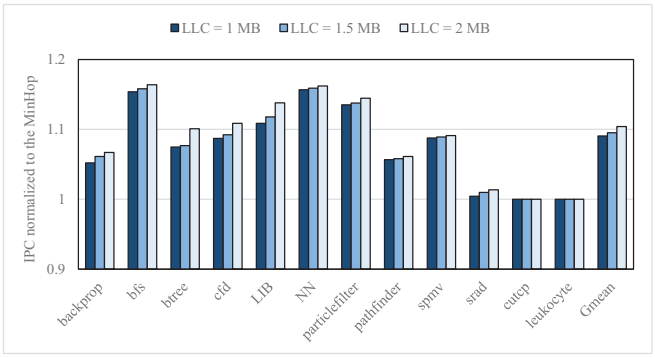

Fig. 8. Normalized IPC results of *MinELI* to the *MinHop* for different LLC sizes.

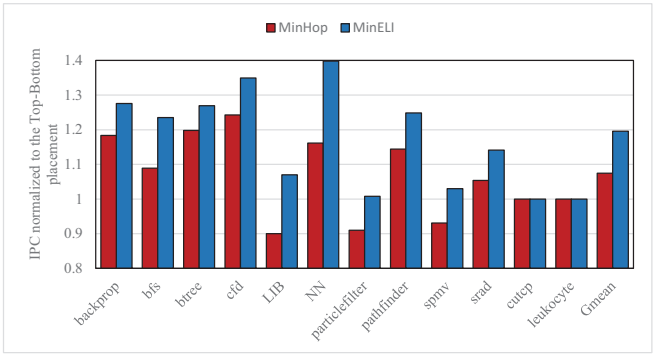

Fig. 9. Normalized IPC results of *MinHop* and *MinELI* with respect to *Top-Bottom* placement (#SM/#LLC = 32/8).

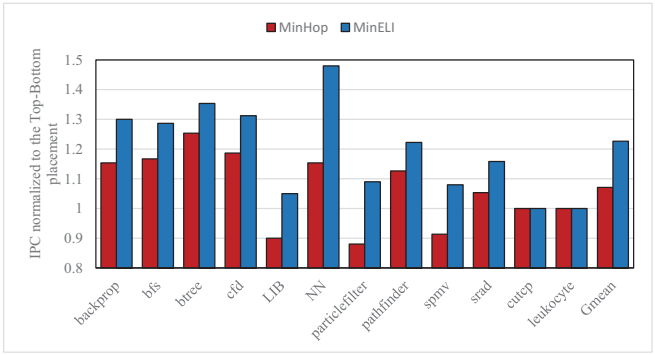

Fig. 10. Normalized IPC results of *MinHop* and *MinELI* with respect to *Top-Bottom* placement (#SM/#LLC = 56/8).

D. Sensitivity analysis

Impact of the LLC size on performance: In this section, we study the impact of the LLC capacity on the performance of *MinELI* placement. To this end, we re-executed the simulations for different LLC sizes (1 MB, 1.5 MB, and 2 MB) and calculated the IPC values for each size. Results are depicted in Figure 8. As shown in this figure, the effect of the LLC capacity on the efficiency of our methodology is negligible when the LLC size is increased from 1 MB to 2 MB.

Impact of the SM/LLC ratio: As explained before, our methodology targets two behaviors of GoNs that they usually operate under near-saturation rates and they have potential hot-spots around the LLC banks. Since these behaviors are

amplified as the ratio of the number of SMs to the number of LLCs increases, we expect that our method performs better with larger ratios. The experimental results for the 24/8 ratio were reported in Figure 7. We have reported the results for the 32/8 and 56/8 ratios in Figures 9 and 10. As can be seen in these figures, the improvements gained by our method increase for higher SM/LLC ratios.

V. Related Work

As mentioned before, reducing the memory access latency is one of the interesting research challenges in GPUs since providing TLP is not sufficient to hide the memory latency [8]. Many GPU resources such as L1 data cache, LLC, on-chip network, and off-chip bandwidth, affect the average memory access latency. Several research studies have attempted to increase the hit rate of GPU caches by increasing their capacity using dense memory cells such as STT-RAM [13] and DWM [16], cache bypassing [10], [2], thread throttling [12], or using data prefetching [8]. On the other hand, several methods have been proposed to improve the performance of GoNs [6], [3], [11], [9]. In this section, we briefly explain the most related method which has been proposed to improve the performance of the GoNs.

Jang et al [6] proposed a bandwidth-efficient on-chip network which results in improving the GPU performance. First, they employed a placement approach which attempts to reduce the average hop count among all of the possible flows. Second, they proposed asymmetric virtual channel partitioning which gives more virtual channels to reply packets (*read-reply* and *write-reply* packets). This is due to the fact that most of the flits travelling over GoN are reply packets. Furthermore, a similar study for the placement of memory controllers in Chip Multi-Processors (CMPs) has been conducted in [1]. In that work, the authors proposed a placement approach which attempts to minimize the maximum channel load. Moreover, they analyzed the effect of routing algorithm on the memory controller placement.

VI. Conclusion

The placement of the LLC banks in the on-chip network is crucial for the GPUs to perform well under network-sensitive workloads. We argued that due to the high network blocking rate in GPUs and their latency hiding capability through TLP, average hop count is not an ideal metric for finding an efficient LLC bank placement. Therefore, we proposed a systematic LLC bank placement approach for GPUs which uses a novel metric, called ELI, and heuristic search algorithm for solving the optimization problem based on ELI. The ELI metric specifically targets the behavior of throughput workloads and hence, assigns higher cost weights to larger latencies as they are more difficult to hide. In order to enable our optimization problem, we defined a synthetic traffic pattern which models the GoN behavior under network-sensitive workloads. We evaluated our proposed placement over the baseline and a placement which attempts to minimize the average hop count of the on-chip network. Experimental results showed that our method outperforms the baseline and the placement with minimum average hop count by up to 29.3% and 15.7% in terms of IPC, respectively.

References

[1] D. Abts, N. D. Enright Jerger, J. Kim, D. Gibson, and M. H. Lipasti, "Achieving predictable performance through better memory controller placement in many-core cmps," in *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ser. ISCA '09. ACM, 2009, pp. 451–461.

[2] R. Ausavarungnirun, S. Ghose, O. Kayiran, G. H. Loh, C. R. Das, M. T. Kandemir, and O. Mutlu, "Exploiting inter-warp heterogeneity to improve gpgpu performance," in *2015 International Conference on Parallel Architecture and Compilation (PACT)*. IEEE, 2015, pp. 25–38.

[3] A. Bakhoda, J. Kim, and T. M. Aamodt, "Throughput-effective on-chip networks for manycore accelerators," in *Proceedings of the 2010 43rd annual IEEE/ACM international symposium on microarchitecture*. IEEE Computer Society, 2010, pp. 421–432.

[4] A. Bakhoda, G. L. Yuan, W. W. Fung, H. Wong, and T. M. Aamodt, "Analyzing cuda workloads using a detailed gpu simulator," in *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 163–174.

[5] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*. IEEE, 2009, pp. 44–54.

[6] H. Jang, J. Kim, P. Gratz, K. H. Yum, and E. J. Kim, "Bandwidth-efficient on-chip interconnect designs for gpgpus," in *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015, p. 9.

[7] N. Jiang, J. Balfour, D. U. Becker, B. Towles, W. J. Dally, G. Michelogiannakis, and J. Kim, "A detailed and flexible cycle-accurate network-on-chip simulator," in *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 86–96.

[8] A. Jog, O. Kayiran, N. Chidambaram Nachiappan, A. K. Mishra, M. T. Kandemir, O. Mutlu, R. Iyer, and C. R. Das, "Owl: Cooperative thread array aware scheduling techniques for improving gpgpu performance," in *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '13. ACM, 2013, pp. 395–406.

[9] H. Kim, J. Kim, W. Seo, Y. Cho, and S. Ryu, "Providing cost-effective on-chip network bandwidth in gpgpus," in *Computer Design (ICCD), 2012 IEEE 30th International Conference on*. IEEE, 2012, pp. 407–412.

[10] C. Li, S. L. Song, H. Dai, A. Sidelnik, S. K. S. Hari, and H. Zhou, "Locality-driven dynamic gpu cache bypassing," in *Proceedings of the 29th ACM on International Conference on Supercomputing*. ACM, 2015, pp. 67–77.

[11] A. Mirhosseini, M. Sadrosadati, M. Zare, and H. Sarbazi-Azad, "Quantifying the difference in resource demand among classic and modern noc workloads," in *ICCD*, 2016.

[12] T. G. Rogers, M. O'Connor, and T. M. Aamodt, "Cache-conscious wavefront scheduling," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-45. IEEE Computer Society, 2012, pp. 72–83.

[13] M. H. Samavatian, H. Abbasitabar, M. Arjomand, and H. Sarbazi-Azad, "An efficient stt-ram last level cache architecture for gpus," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2014, pp. 1–6.

[14] A. Shafiee, M. Zolghadr, M. Arjomand, and H. Sarbazi-Azad, "Application-aware deadlock-free oblivious routing based on extended turn-model," in *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press, 2011, pp. 213–218.

[15] J. A. Stratton, C. Rodrigues, I.-J. Sung, N. Obeid, L.-W. Chang, N. Anssari, G. D. Liu, and W.-m. W. Hwu, "Parboil: A revised benchmark suite for scientific and commercial throughput computing," *Center for Reliable and High-Performance Computing*, vol. 127, 2012.

[16] R. Venkatesan, S. G. Ramasubramanian, S. Venkataramani, K. Roy, and A. Raghunathan, "Stag: spintronic-tape architecture for gpgpu cache hierarchies," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*. IEEE, 2014, pp. 253–264.