MCM-GPU: Multi-Chip-Module GPUs for Continued Performance Scalability - Review

Abdullah Amawi
Computer Network Group
University of Göttingen
Göttingen, Germany
http://www.net.informatik.uni-goettingen.de/

***Abstract*—In this report we will be exploring the importance of GPGPU and how they are crucial for the advancement of many fields in information technology, some of which that are gaining a lot of traction recently such as parallel applications, ranging from scientific computing, data analytics, artificial intelligence, machine learning, and deep learning. Because of the importance of GPU-based high performance computing, improvements on GPUs had to be done on multiple levels from architecture changes to transistors scaling, but as Moore's law slows down and we can no longer scale transistors in historically high rates on a single monolithic GPU die and the need for higher performing GPUs continues, the aforementioned work purposes partitioning the GPU into smaller, easily manufacturable basic GPU Modules (GPMs) and integrating them on package using high bandwidth and power efficient signaling technologies. Moreover it shows how the work resulted into an optimized Multi-Chip-Module GPU (MCM-GPU) design that is 45.5 % faster than the largest implementable monolithic GPU [1].**

This report will be based on the works of Arunkumar et al [1]; The Goal is to give a detailed review of the purposed study and to explore the weaknesses and strengths of it.

***Index Terms*—Graphics Processing Units, Multi-Chip-Modules, Moore's Law.**

I. Introduction

In recent years, big data and data science is increasingly becoming on of the top scientific topics both in information technology research and in industry, which provided and is still providing a lot of advancements and solutions to many real world problems that we were aware of, but we didn't have the compute power to solve, but recently with the emergence of GPUs, GPU-based acceleration became the main propellant in high performance computing (HPC) systems [2], [4], [5], machine learning and data analytics applications in large-scale cloud installations, and personal computing devices [3], [4], [6], [7]. Typically such devices consist of multiple components that include a CPU and one or more GPU accelerators, either way, all of those domains have one thing in common, which is the importance of GPUs and the ability to continuously scale GPU performance [5], [7]. The challenge in all this lies in the fact that in such systems, each GPU has a maximum possible transistor count at the most advanced technology node, and state of the art memory [4]. Until recent years of the continued GPU performance scaling, transistor scaling improved a single GPU performance by increasing the Streaming Multiprocessors (SM) count between GPU generations; Yet, as we noted earlier, the transistor scaling slowed down heavily to a level that Moore's Law no longer applies, which limits the future scaling of single GPU performance and it may result in stopping it [1].

In order to solve the single GPU transistor scaling and the physical development limitations, we can alternatively use multiple GPUs connected on a PCB (Printed Circuit Board), similar to NVIDIA Tesla K10 and K80 [9]. However, the aforementioned work of Arunkumar et al show that it is very hard to scale GPU workloads on such multi-GPU systems, even if they scale much better on a single GPU. There are multiple reasons for this, such as load balancing, work partitioning, and data sharing with the usage of the slow in-board interconnect [1]. But with recent technological advances in packaging [10] and signaling technologies [11], they provide an encouraging integration utilization that would connect the existing on-chip and on-board integration technologies [1].

With those new integration technologies, it results in a new integration tier that Arunkumar et al [1] work want that they utilized to propose a novel **Multi-Chip Module GPU** (MCM-GPU) architecture that will allow us to continue with scaling up GPU performance even with the physical limitations of the manufacturing process and the slowing down of the transistor scaling, and the maximum die size upper limit. They propose that an MCM-GPU should consist of multiple GPU modules (GPM), illustrated in Fig. 1. They state that the basic MCM-GPU architecture will utilize NVIDIA's Ground Reference Signaling (GRS) [11].

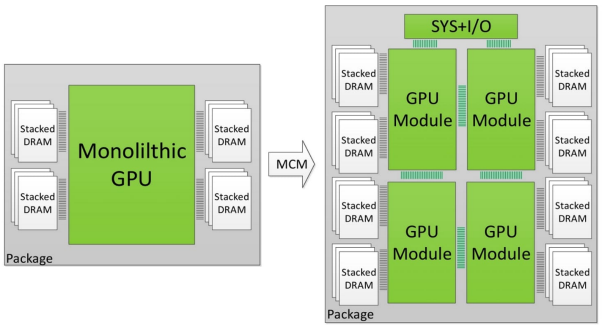

Fig. 1. MCM-GPU: Multiple GPMs and DRAM on a single package [1]

The work of Arunkumar et al first detailed the basic MCM-GPU architecture, then they optimized this proposed MCM-GPU design with multiple innovative design to improve it over the proposed basic MCM-GPU, which are: (i) hardware caches to capture remote traffic in the local GPM [1], (ii) distributed and batched co-operative thread array (CTA) scheduling to better leverage inter CTA locality within a GPM [1], and (iii) first touch page allocation policy to minimize inter-GPM traffic [1]. The aforementioned work claims a couple of points that we will review that they contributed, summarized as:

- They further motivate the continuing need for increasingly powerful GPUs, they argued that by illustrating that many of today's GPU application scale very well with the increasing number of SMs [1]. Moreover, since they demonstrated that the increase in SMs in monolithic GPUs will extremely slow down to a level that it may halt, they propose an MCM-GPU that will continue the scaling beyond what is possible with a single GPU [1].
- They presented a modular MCM-GPU with 256 SMs and discussed its various integration architecture changes, its memory, and signaling technology, and they evaluate them in comparison with the single monolithic GPU, and they show the results of such comparisons both analytically and with simulations [1].
- They propose an MCM-GPU architecture that is both locality-aware and is also better suited to its NUMA (Non-Uniform Memory Access) nature, they also enhanced this architecture to lower the penalty introduced by this non-uniform memory access.
- They also compared the proposed MCM-GPU to another approach which is multi-GPU to confirm that their proposed approach MCM-GPU has advantages to use [1].

II. Motivation and Background

In the recent years, the usage modern graphical processing units to accelerate a big range of parallel applications such as scientific computing, data analytics, and machine learning made this increasing usage also increase the demand for high performing GPUs, which directed the GPU development to have the trend of always increasing the number of streaming multiprocessors (SMs) along with more GPU memory, and the number of transistors which made us realize that there is a physical limit in the size of a GPU, and the scalability when increasing the number of SMs. Table 1 demonstrates the increase of streaming multiprocessors (SMs), memory bandwidth, transistor count, cache, manufacturing node used, and chip size to show this trend [8].

TABLE I
TABLE 1: KEY CHARACTERISTICS OF RECENT NVIDIA GPUS [8].

	Ferm	Kepler	Maxwell	Pascal
SMs	16	15	24	56
BW (GB/s)	177	288	288	720
L2 (KB)	768	1536	3072	4069
Transistors (B)	3.0	7.1	8.0	15.3
Tech. node (mm)	40	28	28	16
Chip size (mm2)	529	551	601	610

A. GPU Application Scalability

In this section we will discuss the GPU scalability and how we benefit from increasing the number of GPU SMs according to Arunkumar et al [1]. It is important to note that both L2 cache and the DRAM bandwidth capacities scale up consistently with the SMs count, such as 384 GB/s for a 32-SM GPU and 3 TB/s for a 256-SM GPU [1]. Moreover, they also demonstrated that increasing the SM count results in different behaviors based on the applications, those applications were classified into two classes based on the parallelism, either high parallelism applications, or limited parallelism applications, they tested 48 different applications, of which 15 of the total were limited parallelism applications that had a limited performance increase with the increase of the SM count, meaning that the performance increase of these limited parallelism applications will flatten out and stop increasing with the increase the SM count. On the other hand, the high parallelism applications, 33 of the 48 total applications tested showed a high level of increase in performance in accordance with the increase of the SM count, they also full utilize the 256-SM GPU [1]. Even though that high level parallelism applications scale very well with increasing SMs, but the problem that we face here is that this increase of SMs at some point is not possible to build with a single monolithic GPU, due to the fact that we mentioned earlier which is the slowdown in transistor scaling [12] that would limit the maximum possible size which translates into the maximum possible die size of a chip, which is the GPU in this case. Moreover, the expected maximum size of single die is around 800 mm^2 according to photo-lithography research [13] and works such as Bruce et al (2007) [14] that specify it as the maximum manufacturable size.

In accordance with the previously mentioned works regarding the photo-lithography and the maximum manufacturable die size the reviewed work of Arunkumar et al [1] assumes that GPUs with more than 128 SMs are not manufacturable on a monolithic die, and they illustrate the performance of such an unmanufacturable with dotted lines in Fig. 2 [1] which also compares the performance scaling of both high parallelism applications and limited parallelism applications.

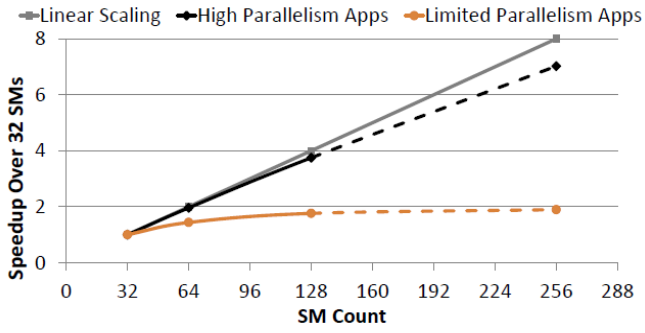

Fig. 2. Hypothetical GPU performance scaling with an increasing number of SMs and memory. 48 applications are considered 33 of which that have enough parallelism to utilize a 256 SMs GPU fully, and 15 that do not. [1]

TABLE II
BANDWIDTH AND ENERGY FOR DIFFERENT INTEGRATIONS (APPROX.) [1].

	Chip	Package	Board	System
BW	10s TB/s	1.5 TB/s	256 GB/s	12.5 GB/s
Energy	80 fJ/bit	0.5 pJ/bit	10 pJ/bit	250 pJ/bit
Overhead	Low	Medium	High	Very High

B. Multi-GPU Alternative

There is also another alternative approach to try to go around the issues of scaling transistors and die size in a single monolithic GPU, which is increasing the application performance by connecting multiple maximally sized monolithic GPUs together into a multi-GPU system, while the idea is simple and can be done, unfortunately it presents multiple new challenges that may prove to be too challenging according to the reviewed work of Arunkumar et al [1]. For example, work distribution poses a challenge and requires significant programming expertise [15]–[20]. Automated multi-GPU software also faces challenges with load balancing, work partitioning, and synchronization [21], [22]. There are even further more unsolved challenges that render this approach to be a worse alternative that are detailed in Arunkumar et al [1], which is mainly surrounding interconnections, off-board, and off-node communications.

C. Package-Level Integration

According to the reviewed work of Arunkumar et al [1], the problem with the previous package-level integration is that they were not suitable for such a design like their proposed MCM-GPU, but the recent advances in this technology, coupled with the addition of the advances with the signaling technologies embodied in NVIDIA's Ground-Referenced Signaling (GRS), this makes the MCM-GPU design possible. Moreover GRS can support up to 20 Gb/s on very low power consumption in comparison to previous technologies [11].

Table 2 demonstrates the different integration domains using the multiple new technological advances that makes the integration of MCM-GPUs possible, also demonstrates the bandwidth, energy consumption, and the relative overhead [1].

III. MULTI-CHIP-MODULE GPUS IDEA AND ANALYSIS.

In this section, we will review the MCM-GPUs idea and their characteristics as shown in the aforementioned work of Arunkumar et al, such as organization and architecture, and other factors. The idea of the MCM-GPU is to combine multiple GPM modules into one single package as mentioned earlier, instead of what it is done today by building one single monolithic GPU, the new approach of MCM-GPUs will allow us to combine smaller, efficient GPM modules into a bigger, better MCM-GPU that can have the potential of adding up to 4x the number of SMs and 2x the memory bandwidth in comparison to the largest single GPU that is compared by this aforementioned paper.

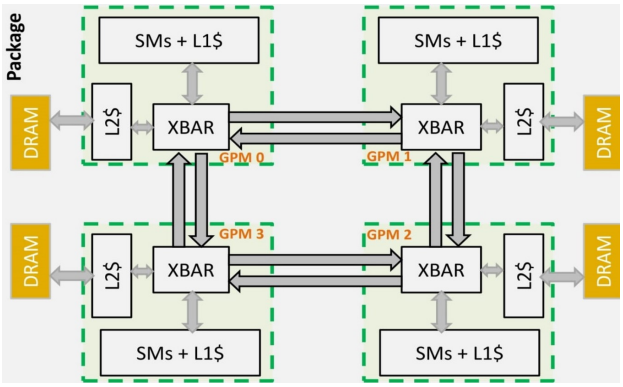

Fig. 3. Basic MCM-GPU of four GPMs [1]

A. MCM-GPU Organization

In the works of Arunkumar et al, the proposed organization of the MCM-GPU is the combination of multiple GPMs that are put together in a single die and combined with a shared I/O and memory, this approach not only will allow us physically to build more capable GPUs without being problematic with the photo-lithography technology and transistor scaling, but also by doing so, it will make it much easier to programmers to utilize, since it will show to the software as a single monolithic GPU that will avoid the steep programming difficulties that were presented earlier with the multi-GPU approach.

B. MCM-GPU and GPM Architecture

Fig. 3 illustrates the architecture of the MCM-GPU that compromises of the following:

- Multiple GPMs, each with it's own SMs, cache, and XBAR
- XBAR is the component that provides communication to the adjacent GPMs.
- Each GPM has it's local DRAM partition but all of them provide globally shared address space.
- The memory system is a Non Uniform Memory Access(NUMA) since GPMs aren't expected to provide full DRAM bandwidth to each other GPM.

All this allows us to build a much larger GPU, such as a GPU with 256 SMs and it will be compromised with smaller GPMs that are are significantly more cost-effective [23], more powerful, while maintaining a smaller overall size.

C. On-Package Bandwidth Considerations

The paper also covered other considerations to optimize the on-package bandwidth such as estimating the on-package bandwidth requirements, performance sensitivity to on-package Bandwidth, and on-package link bandwidth configuration. To demonstrate this, Fig. 4 shows the performance sensitivity in the links of an MCM-GPU compromised of a 4-GPM, and it groups the applications into high parallelism and low parallelism, moreover, the low parallelism is split into memory intensive (M-Intensive) and compute intensive (C-Intensive) applications to see how differently they perform.

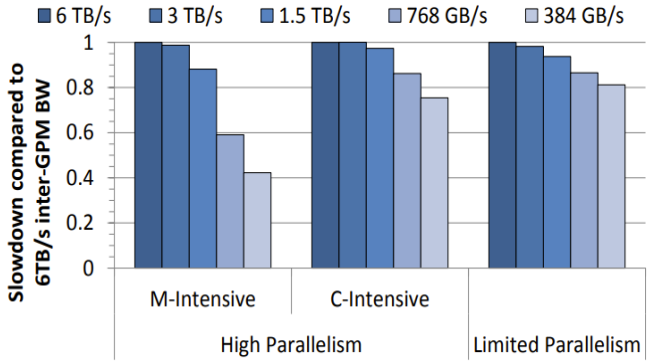

Fig. 4. Performance sensitivity in relation to link bandwidth [1]

IV. Simulation Methodology

The simulation methodology of the aforementioned paper was conducted with the usage of an NVIDIA in-house simulator, the GPU was modeled similar to the recent NVIDIA Pascal GPU [4]. The simulated MCM-GPU software based cache coherence is similar to the Pascal architecture, and the cache itself is compromised of a L1 private cache per SM and a shared L2 cache, all of this is summarized in table 3.

When it comes to the benchmarks that were used in Arunkumar et al work, multiple benchmarks were evaluated that were classified in the following four categories:

- Production class HPC benchmarks from the CORAL benchmarks [24].
- Graph applications from Lonestar suite [25].
- Compute applications from Rodinia [26].
- NVIDIA in-house CUDA benchmarks

The total of the studied benchmark suites are four, but the underlying actual benchmarks were a total of 48 among those different four benchmark suites from CORAL, Lonestar, Rodinia, and NVIDIA. The applications that were utilized represent a big range of GPU application domains that would reflect multiple computing domains that GPUs are heavily used in; such as artificial intelligence, machine learning, deep learning, medical imaging, scientific computing and simulations, graph search, and so on. It is important to note that the used applications were classified by the authors into the two categories as before, high parallelism applications, which have parallel efficiency of equal to or more than 25% and limited parallelism applications which on the other hand have less than 25% parallel efficiency. Moreover as we saw earlier in the performance sensitivity of inter-GPM bandwidth, the high parallelism applications were further classified into memory intensive (M-Intensive) and compute-intensive (C-Intensive) applications. Memory intensive applications can be classified as such if the application suffers more than 20% performance degradation if the memory bandwidth of the system is lowered in half [1]. all simulated benchmarks were ran for one billion wrap instructions, or to completion, any of which occurs earlier. They also presented detailed results for the memory intensive applications to demonstrate the memory footprint per benchmark which are shown in Table 4.

TABLE III
Baseline MCM-GPU configuration [1].

Number of GPMs	4
Total number of SMs.	256
GPU frequency	1GHz
Max number of wraps	64 per SM
Wrap scheduler	Greedy then Round Robin
L1 data cache	128 KB per SM, 128B lines, 4 ways
Total L2 cache	16MB, 128B lines, 16 ways
Inter-GPM interconnect	768GB/s per link, Ring, 32 cycles/hop
Total DRAM bandwidth	3 TB/s
DRAM latency	100ns

TABLE IV
High parallelism, memory intensive applications and their memory usage [1]

Benchmark	Abbr.	Memory Footprint (MB)
Algebraic multigrid solver	AMG	5430
Neural Network Convolution	NN-Conv	496
Breadth First Search	BFS	37
CFD Euler3D	CFD	25
Classic Molecular Dynamics	CoMD	385
Kmeans clustering	Kmeans	216
Lulesh (size 150)	lulesh1	1891
Lulesh (size 190)	Lulesh2	4309
Lulesh unstructured	Lulesh3	203
Adaptive Mesh Refinement	MiniAMR	5407
Mini Contact Solid Mechanics	MnCtct	251
Minimum Spanning Tree	MST	73
Nekbone solver (size 18)	Nekbone1	1746
Nekbone solver (size 12)	Nekbone2	287
SRAD (v2)	Srad-v2	96
Shortest path	SSSP	37
Stream Triad	Stream	3072

V. Optimized MCM-GPU

In this section we will summarize the optimizations made in the paper on their own MCM-GPU.

Introduction of L1.5 cache that is situated between L1 and L2 as the name suggests, the idea is to store remote data made by the correlating GPM partition. Meaning that local memory access will bypass L1.5 cache to reduce remote data access latency and inter-GPM bandwidth overhead [1]. They also evaluated different L1.5 cache size implementations. Starting with 8MB L1.5 cache, a 16MB L1.5 cache which moves most of the L2 to the L1.5 cache, and a 32MB L1.5 cache that only needs to move almost all the L2 cache to the L1.5 side, but also to add another 16MB of cache capacity. all three configurations increase the performance by 4% for 8MB cache, 8% for 16MB cache, and finally 18.3% for L1.5 32MB cache, but they choose the second variation of adding 16MB cache since adding 32MB cache will result in higher performance but it will also cost double the transistor budget [1]. Different configurations and corresponding performance are shown in Fig. 5 and Fig. 6 will demonstrate the addition of L1.5 cache and its position.

Utilizing distributed CTA (Cooperative Thread Array) instead of centralized CTA scheduler in the optimized MCM-GPU, for scheduling GPM locality, which means, instead of splitting consecutive CTAs on different GPM modules, the consecutive CTAs get executed on the same GPM module.

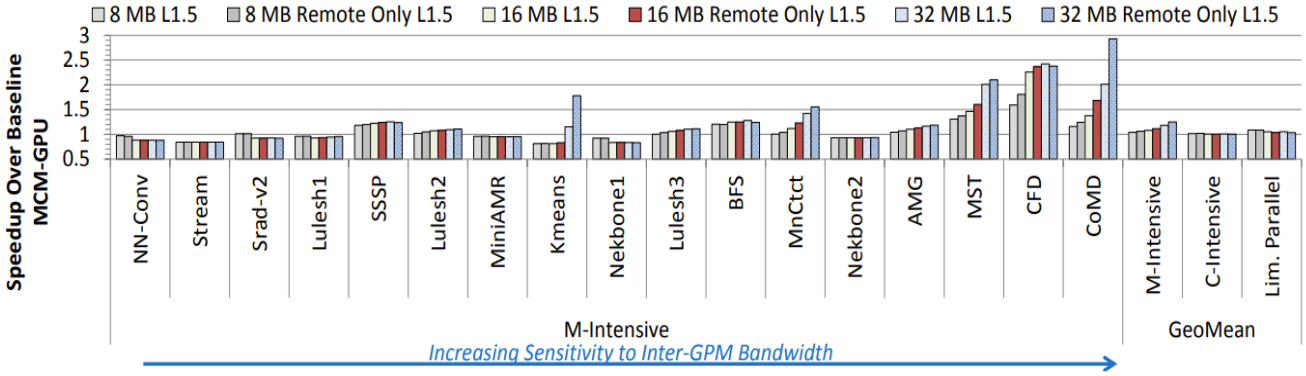

Fig. 5. MCM-GPU with 8MB, 16MB, and 32MB(with added 16MB) L1.5 cache. Memory intensive applications sorted by their inter-GPM bandwidth sensitivity [1].

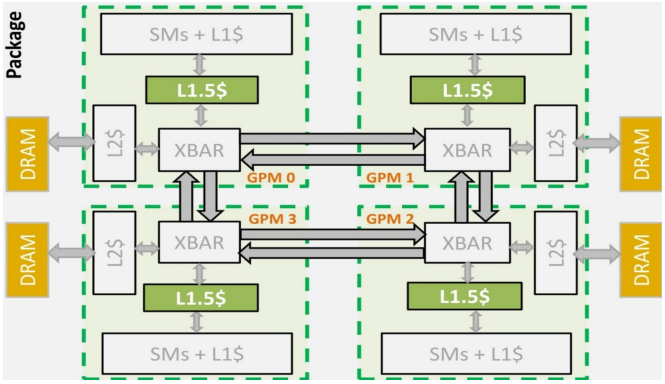

Fig. 6. Optimized MCM-GPU of four GPM and L1.5 cache [1].

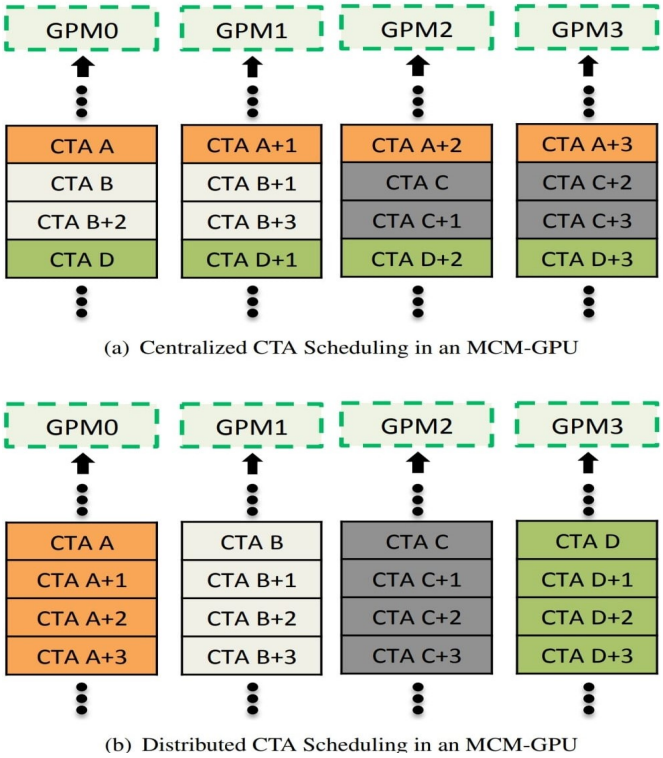

Fig. 7. Optimized MCM-GPU of four GPM and L1.5 cache [1].

To illustrate this, Fig. 7(a) shows that without using the distributed scheduler, this could lead to related CTAs to be split among different GPM modules, but instead with using distributed scheduler, Fig. 7(b) shows that they could enjoy data locality by those related CTAs to be assigned to the same CTA; which also enables better L1.5 cache hit rates and reduce the extra unneeded inter-GPM communication.

First touch page mapping is a policy the authors employed in their optimized MCM-GPU and according to their paper policy maps a page that is referenced for the first time to the local memory partition (MP) to the GPM that the reference is from. An important note about first touch mapping is that it works best alongside the CTA distributed scheduling described previously as a proposed modification by the authors, FT also allows us to use the cache more effectively, and reduces the pressure on the L1.5 cache since as we described earlier that the L1.5 cache is used for remote only access, and the FT policy keeps the access local to each GPM. According to the authors, they claim that on average, the optimized MCM-GPU reduces unneeded inter-GPM communication 5 time in comparison to the base MCM-GPU.

VI. Performance Summary

In this section, we will summarize the performance of the optimized MCM-GPU that is proposed by the reviewed work of Arunkumar et al, to show and summarize both the improvements in the optimized MCM-GPU results and to show to which degree the MCM-GPU idea solved our physical limitation problem of a single monolithic GPU. To summarize the performance increase of the proposed MCM-GPU, memory intensive applications up to 3.2X speedup in CFD were achieved, and 3.5X speed up was achieved in CoMD (Classic Molecular Dynamics). On the other hand, when it comes to compute intensive applications SP (Shortest Path) achieved up to 4.4X speedup and XSBench achieved 3.1X speedup. On the downside, we have to note that some of the applications that have limited parallelism suffered from a performance loss.

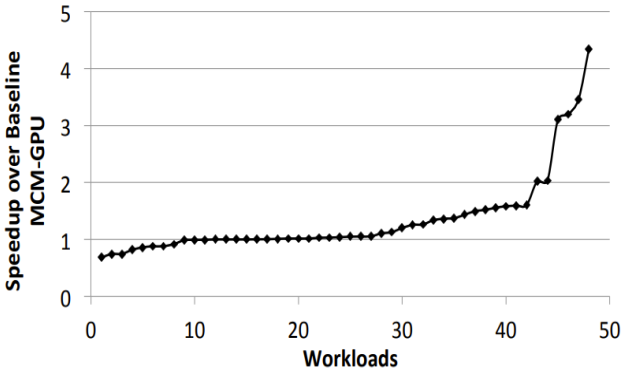

Fig. 8. Performance speedup summary for all workloads [1].

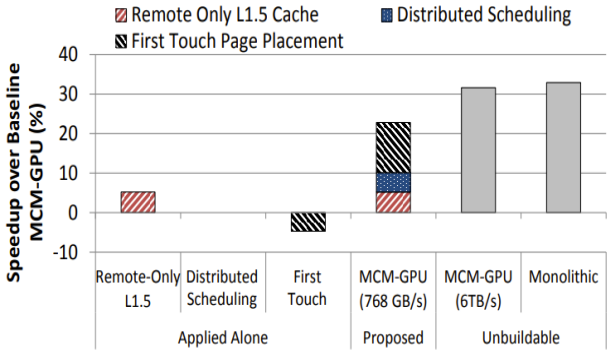

Fig. 9. Performance speedup summary for all workloads [1].

Fig. 8 illustrates the speedups of up to 4.4X as mentioned, but also illustrates the speed ups of all the tested workloads, on the other hand it also shows a few of the applications that suffered from a performance decrease due to having limited parallelism that does not utilize the full power of the MCM-GPU. Fig. 9 demonstrates the three major architectural improvements that makes up the optimized MCM-GPU, it is important to note that as seen in the figure, those optimizations have different behaviors if they are applied alone or together, for example, when remote-only L1.5 is applied, we note a performance increase, but when first touch is applied alone, we can see a performance degradation, on the other hand, applying the distributed scheduling alone has no effect in either direction. Fig. 9 also compares the performance on those improvements combined with hypothetical unbuildable variants of MCM-GPu and an unbuildable monolithic GPU. We observe that we have to apply all three architectural improvements all together to get a speedup of 22.8% as shown.

VII. Multi-GPU Alternative

Another way to increase GPU performance, is made by building multi-GPU systems instead of an MCM-GPU, and we will summarize the comparison made in the aforementioned work between multi-GPU and MCM-GPU just to see why would we pick one over the other, and why the authors chose

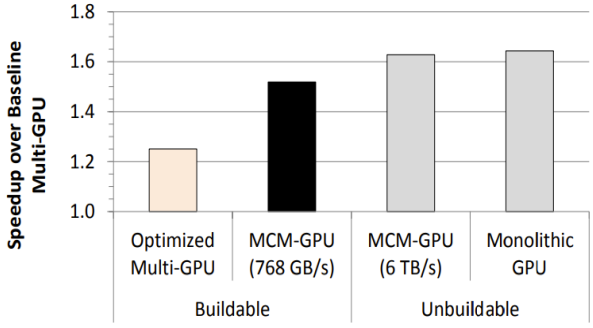

Fig. 10. Performance of Multi-GPU and MCM-GPU [1].

to implement their proposed MCM-GPU knowing that multi-GPU systems existed. Similar to the proposed MCM-GPU that is compromised from 256 SMs, a similar GPU can be built by interconnecting two monolithic GPUs that have 128 SMs each, but the aforementioned work of Arunkumar et al also assumed that to make it a comparable scenario, this multi-GPU should have two features that are a unified memory architecture and an ecosystem of software and hardware that makes it possible to automatically distributes CTAs across the two GPUs to achieve transparency to the programmers [1]. Even then, while assuming those features, multi-GPUs still have many challenges that we mentioned earlier, such as data placement, interconnection bandwidth, workload distribution, and load balancing. Fig. 10 compares this hypothetical optimized multi-GPU with the proposed MCM-GPU in order to get an idea if it has a speedup advantage, knowing that it has many added challenges which are disadvantages in comparison to the MCM-GPU. Fig. 10 illustrates that an optimized multi-GPU that has the added features proposed by the authors to negate some of the downsides of a typical multi-GPU system and to add transparency for the programmers outperforms a typical baseline multi-GPU by an average of 25.1%, but even then, this is not enough to match the very powerful optimized MCM-GPU that outperforms both the optimized multi-GPU and outperforms the baseline multi-GPU by 51.9% on average, the authors contribute this big difference in performance due to the superior quality on-package interconnect that is utilized in the MCM-GPU [1].

VIII. Related Work

While Multi-Chip-Modules have been researched and used heavily in the industry, many of the industry giants utilized this technique, but it was either applied to integrate CPUs modules, such as AMD Opteron 6300 [27] or other components alongside CPUs. also there was the barrier of the signaling technology, unlike the usage of the NVIDIA Ground-Referenced Signaling (GRS) that provides 20 Gbps in comparison to 3.2 Gbps in older technologies [11], or utilizing multi-GPU systems [15], [18], [21], [28]. Table 5 will summarize the related works with the related area, representative article and the advantage of the work of Arunkumar et al in comparison.

TABLE V
RELATED WORKS COMPARISON

Related Areas	Representative Article.	MCM Advantage.
MCM-Design.	Xenos: XBOX360 GPU [29]. The Xeon X5365 [30]. IBM zEnterprise 196 Technical Guide [31]. AMD Server Solutions Playbook [27]. IBM Power Systems Deep Dive [32]. The Compute Architecture of Intel Processor Graphics Gen8 [33].	Only applied to CPU. Combines CPU and GPU on chip
Multi-GPU-Systems.	Memory Access Patterns: The Missing Piece of the multi-GPU Puzzle [34]. Automatic Parallelization of Kernels in Shared-Memory Multi-GPU Nodes [21]. Achieving a Single Compute Device Image in OpenCL for Multiple GPUs [18]. Transparent CPU-GPU Collaboration for Data-parallel Kernels on Heterogeneous Systems [28].	Only work that is fully-suitable for MCM-GPUs. Only work that propose MCM-GPU-as a single logical GPU.
Signaling Tech	The 3rd generation of IBM's elastic interface on POWER6 [35]. Enabling Interposer-based Disintegration of Multi-core Processors [23]. A scalable 0.128-to-1Tb/s 0.8-to-2.6pJ/b 64-lane parallel I/O in 32nm CMOS [36]. A 14-mW 6.25-Gb/s Transceiver in 90-nm CMOS [37]. Ground-Referenced Single-Ended Short-Reach Serial Link in 28 nm CMOS for Advanced Packaging Applications [11].	Operates at up to 3.2 Gbps Vs 20 Gbps Nvidia GRS(Ground-Referenced-Signaling)

IX. OPINION ABOUT PAPER

While reviewing the work of Arunkumar et al. [1], we identified the following strengths and weaknesses in the paper, that we will note in this section.

A. Strengths

While reviewing the aforementioned work, we can clearly identify a couple of major strong points when it comes to the paper, we think that the most important point in here is that the idea of the paper is a novel idea, that actually could be the future of the GPU industry, not only because it surpasses the challenges presented by multi-GPU systems but also the physical limitations of the photo-lithography technology and solves two problems and making a new idea that not only solve this, but could advance the GPU technology greatly. On the other hand, the paper also had clear presentation of the idea, and also explained alternatives such as multi-GPU systems, which gives the reader a complete image and an important background. Moreover, the presented results show the validity of the proposed MCM-GPU clearly, and not only that, but also even illustrates and compares the results with an unbuildable hypothetical single monolithic GPU for the sake of comparison, which shows that the MCM-GPU is not only a solution to a problem, but also an advancement in GPU technology.

B. Weaknesses

Unfortunately, even though the paper is a great addition to GPU research, there are also some major downsides presented in the paper when it comes to multiple points, such as having biased false claims, we think it assumes the end of node technology prematurely, and it ignores the technological advances in GPUs by AMD (Advanced Micro Devices), which is NVIDIA's main competitor in the discrete GPU space. To elaborate on those claims that we make in the review of the paper, AMD in 2017 already had its AMD VEGA architecture, which used a 7 nm process from TSMC (Taiwan Semiconductor Manufacturing Company) [38], moreover, 10nm were announced by Intel corporation in march 28, 2017 [39], before the release of the reviewed work. Meaning that the reviewed work was based on NVIDIA Pascal architecture which is based on TSMC 16 nm [4], which has an older manufacturing process than what is the latest available at the time of the work, so they should have not assumed the end of node technology prematurely. Another point is that huge monolithic GPUs already existed from NVIDIA itself, the Tesla V100 had a die size of 815 mm2 in may 2017 [40].

X. CONCLUSION AND FUTURE WORK

While reviewing the works of Arunkumar et al. [1], we conclude that the approach of solving the limitations of the photo-lithography and the slowdown of transistor scaling when it comes to building bigger, faster, and more capable GPUs lies in the proposed MCM-GPU, so we totally agree with the approach, and we see the benefits of using such an architecture that the authors clearly presented alongside how capable it could be. The paper also summarizes that the proposed MCM-GPU of a 256 SMs U achieves 45.5% speedup more than the largest possible monolithic single GPU with 128 SMs [1]. Moreover, the comparison of the MCM-GPU against the multi-GPU systems not only shows that the MCM-GPU is faster, but also has a lot of advantages against the multi-GPU system that is transparent to the programmers and can be dealt with as a single GPU, unlike the multi-GPU systems that would introduce multiple difficulties that we do not have to deal with in the MCM-GPU.

Future Work. While we believe that works of Arunkumar et al. [1] is basis and the correct direction in GPU technology future, we do believe that similar future works could benefit from utilizing it, but work on the weaknesses we mentioned earlier that could lead into further breakthroughs, mainly by exploring what other GPU architectures could offer to this MCM-GPU design, not only NVIDIA Pascal architecture. On the other hand, keeping an open mind and research this MCM-GPU approach not only inside NVIDIA's proprietary closed ecosystem could also reap some additional benefits, and could lead the to utilize newer photo-lithography technology by multiple sources, not only what NVIDIA is utilizing.

REFERENCES

[1] A. Arunkumar, E. Bolotin, B.Cho, U.Milic, E.Ebrahimi, O.Villa, A.Jaleel, C.Wu and D. Nellans, "MCM-GPU: Multi-Chip-Module GPUs for Continued Performance Scalability" 2017.

[2] 2015. TOP500 Shows Growing Momentum for Accelerators. (2015). http://insidehpc.com/2015/11/top500-shows-growing-momentum-for-accelerators/ Accessed:2016-06-20.

[3] 2016. NVIDIA cuDNN, GPU Accelerated Deep Learning. (2016). https://developer.nvidia.com/cudnn Accessed: 2016-11-17.

[4] 2016. The New NVIDIA Pascal Architecture. (2016). http://www.nvidia.com/object/gpu-architecture.html Accessed: 2016-06-20.

[5] Michael Feldman, Christopher G. Willard, and Addison Snell. 2015. HPC Application Support for GPU Computing. (2015). http://www.intersect360.com/ industry/reports.php?id=131

[6] Andrew Lavin and Scott Gray. 2016. Fast Algorithms for Convolutional Neural Networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR '16). IEEE, Las Vegas, NV, USA, 4013–4021. https://doi.org/10.1109/CVPR.2016.435

[7] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. ArXiv e-prints (Sept. 2014). arXiv:cs.CV/1409.1556

[8] 2016. Inside Pascal: NVIDIA's Newest Computing Platform. (2016). https://devblogs.nvidia.com/parallelforall/inside-pascal Accessed: 2016-06-20.

[9] 2015. TESLA K80 GPU ACCELERATOR . (2015). https://images.nvidia. com/content/pdf/kepler/Tesla-K80-BoardSpec-07317-001-v05.pdf Accessed: 2016-06-20.

[10] Mitsuya Ishida. 2014. Kyocera APX - An Advanced Organic Technology for 2.5D Interposers. (2014). https://www.ectc.net Accessed: 2016-06-20.

[11] John W. Poulton, William J. Dally, Xi Chen, John G. Eyles, Thomas H. Greer, Stephen G. Tell, John M. Wilson, and C. Thomas Gray. 2013. A 0.54 pJ/b 20Gb/s Ground-Referenced Single-Ended Short-Reach Serial Link in 28 nm CMOSfor Advanced Packaging Applications. IEEE Journal of Solid-State Circuits 48, 12 (Dec 2013), 3206–3218. https://doi.org/10.1109/JSSC.2013.2279053

[12] 2015. International Technology Roadmap for Semiconductors 2.0. (2015). http://www.itrs2.net/itrs-reports.html

[13] 2016. The TWINSCAN NXT:1950i Dual-Stage Immersion Lithography System. (2016). https://www.asml.com/products/systems/twinscan-nxt/twinscan-nxt1950i/en/s46772?dfp_product_id=822 Accessed: 2016-11-18.

[14] Bruce W. Smith and Kazuaki Suzuki. 2007. Microlithography: Science and Technology, Second Edition.https://books.google.com/books?id=_hTLDCeIYxoC

[15] Tal Ben-Nun, Ely Levy, Amnon Barak, and Eri Rubin. 2015. Memory Access Patterns: The Missing Piece of the multi-GPU Puzzle. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '15). ACM, New York, NY, USA, 19:1–19:12. https://doi.org/ 10.1145/2807591.2807611

[16] Long Chen, Oreste Villa, and Guang R. Gao. 2011. Exploring Fine-Grained Task-Based Execution on Multi-GPU Systems. In Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER '11). IEEE, Washington, DC, USA, 386–394. https://doi.org/10.1109/CLUSTER.2011.50

[17] Long Chen, Oreste Villa, Sriram Krishnamoorthy, and Guang R. Gao. 2010. Dynamic load balancing on single- and multi-GPU systems. In Proceedings of the IEEE International Symposium on Parallel Distributed Processing (IPDPS '10). IEEE, Atlanta, GA, USA, 1–12. https://doi.org/10.1109/IPDPS.2010.5470413

[18] Jungwon Kim, Honggyu Kim, Joo Hwan Lee, and Jaejin Lee. 2011. Achieving a Single Compute Device Image in OpenCL for Multiple GPUs. In Proceedings of the 16th ACM Symposium on Principles and Practice of Parallel Programming (PPoPP '11). ACM, New York, NY, USA, 277–288. https://doi.org/10.1145/ 1941553.1941591

[19] Takuji Mitsuishi, Jun Suzuki, Yuki Hayashi, Masaki Kan, and Hideharu Amano. 2016. Breadth First Search on Cost-efficient Multi-GPU Systems. SIGARCH Comput. Archit. News 43, 4 (April 2016), 58–63. https://doi.org/10.1145/2927964. 2927975

[20] Jeff A. Stuart and John D. Owens. 2011. Multi-GPU MapReduce on GPU Clusters. In Proceedings of the IEEE International Parallel & Distributed Processing Symposium (IPDPS '11). IEEE, Washington, DC, USA, 1068–1079. https://doi. org/10.1109/IPDPS.2011.102

[21] Javier Cabezas, Lluís Vilanova, Isaac Gelado, Thomas B. Jablin, Nacho Navarro, and Wen-mei W. Hwu. 2015. Automatic Parallelization of Kernels in SharedMemory Multi-GPU Nodes. In Proceedings of the 29th ACM on International Conference on Supercomputing (ICS '15). ACM, New York, NY, USA, 3–13. https://doi.org/10.1145/2751205.2751218

[22] Jeff A. Stuart and John D. Owens. 2009. Message Passing on Data-parallel Architectures. In Proceedings of the IEEE International Symposium on Parallel & Distributed Processing (IPDPS '09). IEEE, Washington, DC, USA, 1–12. https://doi.org/10.1109/IPDPS.2009.5161065

[23] Ajaykumar Kannan, Natalie Enright Jerger, and Gabriel H. Loh. 2015. Enabling Interposer-based Disintegration of Multi-core Processors. In Proceedings of the 48th International Symposium on Microarchitecture (MICRO-48). ACM, New York, NY, USA, 546–558. https://doi.org/10.1145/2830772.2830808

[24] 2014. CORAL Benchmarks. (2014). https://asc.llnl.gov/CORAL-benchmarks/

[25] Molly A. O'Neil and Martin Burtscher. 2014. Microarchitectural performance characterization of irregular GPU kernels. In Proceedings of the IEEE International Symposium on Workload Characterization (IISWC '14). IEEE, Raleigh, NC, USA, 130–139. https://doi.org/10.1109/IISWC.2014.6983052

[26] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy W. Sheaffer, Sang-Ha Lee, and Kevin Skadron. 2009. Rodinia: A Benchmark Suite for Heterogeneous Computing. In Proceedings of the IEEE International Symposium on Workload Characterization (IISWC '09). IEEE, Washington, DC, USA, 44–54. https://doi.org/10.1109/IISWC.2009.5306797

[27] 2012. AMD Server Solutions Playbook. (2012). http://www.amd.com/Documents/ AMD_Opteron_ServerPlaybook.pdf Accessed: 2016-08-19.

[28] Janghaeng Lee, Mehrzad Samadi, Yongjun Park, and Scott Mahlke. 2013. Transparent CPU-GPU Collaboration for Data-parallel Kernels on Heterogeneous Systems. In Proceedings of the 22Nd International Conference on Parallel Architectures and Compilation Techniques (PACT '13). IEEE, Piscataway, NJ, USA, 245–256. http://dl.acm.org/citation.cfm?id=2523721.2523756

[29] 2005. Xenos: XBOX360 GPU. (2005). http://fileadmin.cs.lth.se/cs/Personal/ Michael_Doggett/talks/eg05-xenos-doggett.pdf Accessed: 2016-08-19.

[30] 2007. The Xeon X5365. (2007). http://ark.intel.com/products/30702/Intel-Xeon-Processor-X5365-8M-Cache-3_00-GHz-1333-MHz-FSB Accessed: 2016-08-19.

[31] 2011. IBM zEnterprise 196 Technical Guide. (2011). http://www.redbooks.ibm. com/redbooks/pdfs/sg247833.pdf Accessed: 2016-08-19.

[32] 2012. IBM Power Systems Deep Dive. (2012). http://www-05.ibm.com/cz/ events/febannouncement2012/pdf/power_architecture.pdf Accessed: 2016-08-19.

[33] 2015. The Compute Architecture of Intel Processor Graphics Gen8. (2015). https://software.intel.com Accessed: 2016-08-19.

[34] Tal Ben-Nun, Ely Levy, Amnon Barak, and Eri Rubin. 2015. Memory Access Patterns: The Missing Piece of the multi-GPU Puzzle. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '15). ACM, New York, NY, USA, 19:1–19:12. https://doi.org/ 10.1145/2807591.2807611

[35] Daniel Dreps. 2007. The 3rd generation of IBM's elastic interface on POWER6. In Proceedings of the IEEE Hot Chips 19 Symposium (HCS '19). IEEE, 1–16. https://doi.org/10.1109/HOTCHIPS.2007.7482489

[36] Mozhgan Mansuri, James E. Jaussi, Joseph T. Kennedy, Tzu-Chien Hsueh, Sudip Shekhar, Ganesh Balamurugan, Frank O'Mahony, Clark Roberts, Randy Mooney, and Bryan Casper. 2013. A scalable 0.128-to-1Tb/s 0.8-to-2.6pJ/b 64-lane parallel I/O in 32nm CMOS. In IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC '13). IEEE, San Francisco, CA, USA, 402–403. https://doi.org/10.1109/ISSCC.2013.6487788

[37] John Poulton, Robert Palmer, Andrew M. Fuller, Trey Greer, John Eyles, William J. Dally, and Mark Horowitz. 2007. A 14-mW 6.25-Gb/s Transceiver in 90-nm CMOS. IEEE Journal of Solid-State Circuits 42, 12 (Dec 2007), 2745–2757. https://doi.org/10.1109/JSSC.2007.908692

[38] https://www.tsmc.com/english

[39] https://newsroom.intel.com/news/intel-presents-technology-manufacturing-day-live-video-updates/#gs.x0me7j

[40] https://developer.nvidia.com/blog/inside-volta/